Vers une solution réaliste de décodage source-canal conjoint de contenus multimédia

Cédric Marin

N° D'ORDRE : 9401

THÈSE DE DOCTORAT

SPECIALITÉ : PHYSIQUE

École Doctorale « Sciences et Technologies de l'Information, des Télécommunications et des Systèmes »

Présentée par :

Cédric MARIN

Sujet :

Vers une solution réaliste de décodage source-canal conjoint de contenus multimédia

Soutenue le 27 Mars 2009 devant les membres du jury :
M. AL AGHA Khaldoun - Professeur à l'Université d'Orsay
M. ANTONINI Marc - Directeur de Recherche au CNRS de Nice
M. DUHAMEL Pierre - Directeur de Recherche au CNRS d'Orsay
M. KIEFFER Michel - Maître de Conférence à l'Université d'Orsay
M. LACAN Jérôme - Maître de Conférence à l'Université de Toulouse
M. SALAMATIAN Kavé - Reader à l'Université de Lancaster (UK)
M. ROUFFET Denis - Ingénieur chez Alcatel Lucent Bell Labs

Remerciements

Mes premiers remerciements sont dédiés à Denis Rouffet qui a été l'initiateur de cette thèse. Il a su me faire confiance et m'accueillir chaleureusement au sein de son équipe de recherche chez Alcatel pendant les trois ans qu'ont duré cette thèse.

Je me tourne ensuite vers mes deux responsables académiques : Pierre Duhamel et Michel Kieffer. Sans eux, ce projet n'aurait jamais pu aboutir. Pierre, un immense merci pour ta disponibilité, ton soutien, ton écoute et tes innombrables conseils. Michel, un grand merci d'abord de m'avoir transmis ta passion pour le traitement du signal puisque c'est toi qui m'as fait découvrir cette matière à la FIUPSO. Un énorme merci ensuite pour ton attention, ta pédagogie, ta bonne humeur, ton positivisme tout au long de cette thèse... et bien sûr, pour ton talent de grand Maître !

Je témoigne également toute ma gratitude aux personnes que j'ai pu côtoyer et avec lesquelles j'ai pu collaborer durant ces trois années chez Alcatel, et plus spécifiquement, à toutes les équipes rattachées à Laurent Thomas. Je ne citerai pas tous les noms car la liste serait trop exhaustive mais je souhaite tout de même nommer le noyau dur : Vinod, Yann, Sylvaine, Marie-Line, Frédéric et Bessem. Je ne saurais vous remercier suffisamment pour tout ce que vous m'avez apporté durant cette aventure.

Je remercie également toutes les personnes du LSS avec lesquelles j'ai pu travailler durant cette thèse et pendant mon année de post-doctorat. Je pense notamment à Khaled, Sofiane, Amadouche, Véronichien, Brice, Ali, Sajad, José, Sophie, Francesca, Patrice, Alex, Samson, Zied, Cagatay, Léonardo, Ziad, Anissa, Salma... On était tous embarqué dans la même galère (la thèse) et ce défi a été bien moins difficile grâce à vous.

J'adresse mes sincères remerciements aux membres du jury qui ont consacré un peu de leur temps à l'étude de mon travail ainsi que pour leurs commentaires très positifs.

Un grand merci également à tous mes amis, et plus spécifiquement, à Joulux, Guillaume, Laurine, Bat, Nadia, Marc pour leur immense soutien et leur amitié sincère.

Mes derniers remerciements s'adressent à ma famille à qui je dédie cette thèse.

Je remercie notamment mes parents de m'avoir donné la possibilité de suivre ces études passionnantes, pour leur soutien sans faille, leur amour et leur encouragement durant toutes ces années.

Je remercie chaleureusement mon cher frère et mes deux adorables soeurs pour leur présence, leur écoute, leurs conseils... et tout le reste !

Enfin, j'exprime mes profonds remerciements à Gaëlle, devenue récemment mon épouse, pour son soutien et ses encouragements depuis le début de nos études. Que ce travail soit la preuve de mon amour.

À mes grands-parents,

à mes parents,

à mon frère et mes soeurs,

à Gaëlle...

Table des matières

Table des figures

Introduction

Dans les prochaines années, les téléphones mobiles vont intégrer des services de plus en plus nombreux basés sur le transport de la vidéo. Actuellement, plusieurs problèmes techniques doivent encore être résolus pour aboutir à une solution optimisée.

Lors d'une transmission vidéo sur les réseaux mobiles, les données doivent être efficacement comprimées pour s'adapter à la bande-passante réduite. Cependant, plus les données multimédia sont comprimées, plus le flux est sensible aux erreurs de transmission. Lors du décodage vidéo, une simple erreur binaire peut entrainer la perte totale d'une séquence d'images. Par conséquent, le flux encodé entrant dans le décodeur vidéo du récepteur ne doit pas être dégradé.

Dans les transmissions radio-mobiles, le signal arrivant sur l'antenne du récepteur est fortement détérioré par les perturbations survenant sur le canal (atténuation du signal radio, interférences multiples, effet Doppler). Pour garantir un signal exempt d'erreur à l'entrée du décodeur vidéo, plusieurs mécanismes de protection sont implémentés dans le récepteur. La première solution consiste à regrouper les données en paquets protégés par un code de détection d'erreurs basé sur un CRC (*Cyclic Redundancy Check*) ou un checksum. Les paquets, dont l'intégrité n'est pas assurée à la réception, peuvent être retransmis (*Automatic Repeat reQuest* ou ARQ) ou effacés. Cependant, ces mécanismes de retransmission peuvent être difficilement intégrés dans des scénarios avec des contraintes de délais importantes (la visiophonie), voir même impossibles dans les situations de diffusion (télévision par satellite).

Pour remédier à ce problème de latence, les solutions standards utilisent des codes robustes de correction d'erreurs, tels que les turbo-codes ou les LDPC (*Low Density Parity Check*), pour corriger les erreurs de transmission. Ce principe consiste à ajouter des redondances dans les paquets transmis et à les utiliser à la réception pour retrouver les données originales. Cependant, les redondances introduites par ces codes sont souvent mal proportionnées. La quantité de redondances est parfois surdimensionnée quand le canal est faiblement perturbé, réduisant la bande-passante allouée aux données. Au contraire, lorsque les conditions radios sont mauvaises, certains paquets erronés peuvent ne pas être corrigés. Les paquets sont alors perdus. Dans de telles situations, des techniques de dissimulation d'erreurs (*error concealment*) peuvent être utilisées dans le décodeur vidéo. Ces mécanismes exploitent les corrélations spatiales et temporelles des images pour estimer les portions manquantes dans la vidéo décodée.

Ces dernières années, des techniques de décodage conjoint source-canal (*Joint Source-Channel Decoding* ou JSCD) ont été proposées pour corriger plus efficacement les paquets erronés. Ces méthodes exploitent les redondances résiduelles contenues dans le flux reçu pour améliorer la qualité de la vidéo décodée. Les redondances résiduelles peuvent être de natures différentes (informations souples, sémantique et syntaxe du train binaire, propriétés de paquétisation, etc.) et ces informations ont un impact variable sur les performances obtenues. Durant cette thèse, nous avons introduit une nouvelle méthode JSCD exploitant à la fois les propriétés sémantiques et syntaxiques du flux vidéo ainsi que le CRC de la couche Liaison. Cette technique a ensuite été testée sur le dernier standard de compression vidéo : le H.264/AVC. Une description détaillée de la méthode ainsi que les résultats expérimentaux sont fournis dans ce rapport. La version du papier revue soumis à *IEEE Transactions on Communications* est donnée dans l'annexe B.

Parallèlement, pour pouvoir intégrer ces outils robustes dans le récepteur, de nombreuses modifications sont nécessaires. Il faut notamment pouvoir faire remonter des paquets contenant des erreurs au niveau du décodeur vidéo (étant donné que les traitements robustes sont implémentés au niveau du décodeur vidéo). Or, comme nous l'avons souligné précédemment, les paquets erronés sont effacés par les mécanismes de protection avant d'avoir atteint le décodeur vidéo. Cette problématique a donc été étudiée dans une deuxième phase et a abouti à un nouveau principe de couche perméable. Cette méthode innovante est implémentée dans chaque couche protocolaire et consiste à désactiver la détection d'erreurs sur les données du paquet. A la place, le code de détection d'erreurs est utilisé comme un code de correction d'erreurs pour corriger les champs importants contenus dans l'en-tête du paquet. Une fois l'en-tête du paquet corrigée, les données transportées (correctes ou incorrectes) peuvent être transmises à la couche supérieure sans risque de perte. En intégrant ce mécanisme dans chaque couche protocolaire, les données vidéo erronées peuvent arriver à l'entrée du décodeur vidéo, où elles sont ensuite traitées par les algorithmes robustes. Une description plus précise de la méthode ainsi que des résultats de simulation sont donnés dans ce rapport. La version du papier revue soumis à *IEEE Transactions on Communications* est donnée dans l'annexe C.

Pour réduire l'étendue de ce sujet, nous avons limité notre étude à la transmission d'un flux vidéo encodé en H.264/AVC entre un serveur et un terminal WiFi connectés à Internet.

L'originalité de ce travail repose en grande partie sur la prise en compte de l'ensemble des processus associés à la transmission lors du décodage robuste de la vidéo. Or, comme nous l'avons déjà mentionné, les algorithmes de décodage robuste exploitent les redondances structurelles du flux reçu pour corriger des erreurs. C'est pourquoi, les premiers chapitres de cette thèse décrivent avec précision la structure fine des flux binaires intervenant au niveau du récepteur.

Organisation de la thèse

Ce rapport est divisé en quatre chapitres qui sont suivis de trois annexes. Une explication brève du contenu de chaque chapitre est donnée ci-dessous :

Le chapitre 1 débute par une introduction au codage source. Il présente de façon détaillée les divers outils intervenant dans un schéma de compression vidéo, et plus spécifiquement, dans la norme H.264/AVC. Cette norme correspond au dernier standard d'encodage vidéo et caractérise notre référence dans ce domaine.

Le chapitre 2 présente l'architecture générale du réseau Internet dans un environnement radio-mobile. Il identifie le maillon faible d'un tel système et désigne les points d'action. Les protocoles de transmission intervenant sur le support radio sont finalement introduits.

Le chapitre 3 commence par présenter les mécanismes associés à la transmission robuste de la vidéo. Il décrit ensuite la nouvelle méthode JSCD exploitant les redondances structurelles du flux vidéo et le CRC contenu dans les fragments de la couche Liaison. Cette technique a été adaptée au décodage des résidus de prédiction contenus dans le flux H.264/AVC. Les résultats des simulations sont finalement introduits et illustrent l'intérêt de cette technique.

Le chapitre 4 traite des problèmes liés à l'intégration des outils JSCD dans le terminal. Il décrit en détail le principe de fonctionnement du nouveau modèle de couche perméable basé sur la correction des en-têtes protocolaires. Les résultats des simulations, réalisées sur les couches PHY et MAC de WiFi, sont finalement exposés. Ils mettent en valeur les avantages apportés par la méthode.

Nous terminons ce rapport par une conclusion générale qui résume les travaux présentés et propose des thèmes de recherche qui pourraient être abordés ultérieurement dans ce domaine.

Publications relatives à mes travaux

Les travaux de recherche réalisés durant cette thèse ont abouti à plusieurs publications qui sont citées ci-dessous.

Articles de revue

C. Marin, K. Bouchireb, M. Kieffer, and P. Duhamel. Joint exploitation of residual source information and MAC layer CRC redundancy for robust video decoding. *IEEE Transactions on Wireless Communications*. Accepté.

C. Marin, Y. Leprovost, M. Kieffer, and P.Duhamel. Robust MAC-Lite and soft header recovery for packetized multimedia transmission. *IEEE Transactions on Communications*. Accepté.

Articles de conférence

K. Bouchireb, C. Marin, P. Duhamel, and M. Kieffer. Improved retransmission scheme for video communication systems. In Proceedings of IEEE PIMRC 08. Pages : 1-5. Cannes, France. September 2008.

C. Marin, Y. Leprovost, M. Kieffer, and P. Duhamel. Robust MAC-Lite and header recovery based improved permeable protocol layer scheme. In Proceedings of IEEE ISSSTA 08. Pages : 496-501. Bologne, Italy. August 2008.

C. Marin, Y. Leprovost, M. Kieffer, and P. Duhamel. Robust header recovery based enhanced permeable protocol layer mechanism. In Proceedings of IEEE SPAWC 08. Pages : 91-95. Recife, Brasil. July 2008.

C. Marin, P. Duhamel, K. Bouchireb, and M. Kieffer. Robust video decoding through simultaneous usage of residual source information and MAC layer CRC redundancy. In Proceedings of IEEE Globecom 07. Pages : 2070-2074. Washington, USA. November 2007.

Brevets

C. Marin, M. Kieffer, and P. Duhamel. Selective recovery of CRC protected data. European patent N° 08305153.2. May 2008.

C. Marin, M. Kieffer, and P. Duhamel. Recovery of transmission errors. European patent N° 08305151.6. May 2008.

C. Marin and Y. Leprovost. Process for delivering a video stream over a wireless channel. European patent N° 07291489.8. December 2007.

C. Marin and Y. Leprovost. Process for delivering a video stream over a wireless bidirectional channel between a video encoder and a video decoder. European patent N° 07291490.6. December 2007.

C. Marin, Y. Leprovost, M. L. Alberi-Morel et S. Kerboeuf. Système de transfert de contenus en couches, en mode broadcast et/ou unicast, à des terminaux mobiles rattachés à un réseau à couverture radio non uniforme et à mécanisme de transfert discontinu de contenus. Brevet français N° 0758450. Octobre 2007.

C. Marin, Y. Leprovost, and V. Kumar. Transmitting data packets over an air interface. European patent N° 07290418.8. April 2007.

Autres publications

C. Marin, S. Ben-Jamaa, M. Kieffer et P. Duhamel. Rapport CNES Cross-Layer - Etat de l'art des outils de codage vidéo. Février 2006.

Chapitre 1

La compression vidéo

Dans ce chapitre, nous décrivons le principe de fonctionnement des codeurs vidéo, et plus particulièrement, du codeur H.264/AVC. La structure fine des flux vidéo générés par ce codeur est également détaillée. Cette description approfondie sera utile dans les chapitres 3 et 4 portant sur les problématiques de transmission robuste.

1.1 Historique

Une présentation détaillée de l'évolution historique des codeurs vidéo peut être trouvée dans [1].

La manière la plus simple de réaliser un codage vidéo consiste à coder chaque image du flux par le biais d'un codeur d'images fixes, par exemple JPEG [2]. Dans ce type de schéma, seules les redondances spatiales des images sont utilisées pour comprimer la vidéo (codage en mode Intra). Cependant, l'idée d'exploiter la corrélation temporelle entre les images successives d'un flux vidéo apparaît dès 1929 [3]. Dans cette approche, seules les différences entre les images successives doivent être transmises au récepteur (codage en mode Inter). Le tout premier standard proposé par l'ITU-T (*International Telecommunications Union - Telecommuniction*), H.120 [4], met en oeuvre cette idée. Dans cette norme, les images sont découpées en blocs. Les blocs identiques à leurs correspondants dans l'image précédente ne sont pas codés et les autres sont traîtés en mode Intra.

Les codeurs de type H.120 sont cependant inefficaces en cas de mouvement d'ensemble de la caméra vis-à-vis de la scène filmée. Pour résoudre ce problème, la solution a été d'exploiter une partie des informations contenues dans l'image précédente pour prédire les blocs de l'image courante et de transmettre des données permettant de corriger cette prédiction [5]. Ce principe a donné naissance aux codeurs vidéo hybrides, dont le nom provient du fait qu'ils utilisent deux techniques de réduction de redondances : d'une part, une prédiction temporelle, d'autre part, une transformation des résidus de prédiction. Cette structure de base a été

formalisée par l'ITU-T dans le standard H.261 [6]. Les standards ultérieurs, MPEG-1 [7], MPEG-2 [8], H.263 [9], MPEG-4 Part 2 Visual [10] et H.264/AVC [11] ont essentiellement repris et amélioré (fortement) cette structure de base de codage hybride.

1.2 Les outils élémentaires pour la compression

Quel que soit le type de codeur (voix, musique, image fixe, vidéo), un certain nombre d'outils communs permettent de réaliser la compression du signal. Ces outils, brièvement décrits ci-dessous et illustrés sur la figure 1.1, sont largement détaillés dans [12].

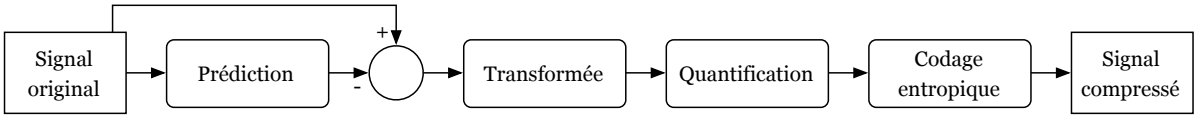


FIG. 1.1 – Schéma basique d'un codeur

La prédiction : ce mécanisme permet d'établir un ensemble de valeurs prédites sur les échantillons du signal d'entrée. Généralement, cette estimation est basée sur les informations des échantillons déjà encodés du signal. La différence entre les échantillons du signal original et leur prédiction permet d'obtenir les résidus de prédiction qui sont plus efficaces à compresser.

La transformation : Ce procédé consiste à projeter les résidus de prédiction dans une base permettant de réduire la corrélation statistique entre les échantillons des résidus de prédiction. Cette opération permet de rassembler l'énergie du signal sur un faible nombre d'échantillons. Une décorrélation optimale des données est obtenue à partir de la transformée de Karhunen-Loève (*Karhunen-Loève Transform* ou KLT) qui projette le signal sur les vecteurs propres de sa matrice de covariance. Cependant, cette transformation est complexe et les codeurs traditionnels intègrent plutôt une décomposition en sous-bandes fréquentielles. Dans cette catégorie, la transformée en cosinus discrète (*Discrete Cosine Transform* ou DCT) est de loin la plus utilisée à la fois pour sa simplicité algorithmique et ses performances [13].

La quantification : Ce procédé permet d'approximer les échantillons du signal transformé afin de réduire la quantité d'information à transmettre. Souvent, la précision de cette approximation est contrôlée à l'aide d'un pas représentant la plus petite valeur absolue représentable, mais également l'incrément entre deux valeurs successives à la sortie du quantificateur. Parmi tous les outils de compression, la quantification est la seule opération à être irréversible, induisant des pertes d'information. L'objectif de tout codeur est de minimiser la perte d'information résultante sous une contrainte de débit en sortie du codeur.

Le codage entropique : Ce mécanisme permet de représenter efficacement les symboles issus du quantificateur en prenant en compte leur fréquence d'apparition. Ainsi, un mot de code de longueur variable (*Variable Length Coding* ou VLC) est associé à chaque symbole en fonction de la statistique de la source. Les mots de code les plus courts sont attribués aux

symboles fréquents, et inversement, des mots de code plus longs sont réservés aux symboles les moins probables. Les codeurs actuels utilisent ce procédé pour encoder les paramètres de compression, le plus connu étant le codage de Huffman.

1.3 La norme H.264/AVC

En comparaison avec les anciens standards de compression vidéo, le H.264/AVC [14] est basé sur une véritable révolution algorithmique qui permet d'atteindre un seuil de codage qui n'était pas prévisible huit ans auparavant. Ces progrès ont été rendus possible par l'union des experts vidéo de l'ITU-T et de MPEG (*Moving Picture Experts Group*) qui ont établi la JVT (*Joint Video Team*) en décembre 2001 afin de finaliser la norme. H.264/AVC fut finalisée en mars 2003 et approuvée par l'ITU-T en mai 2003.

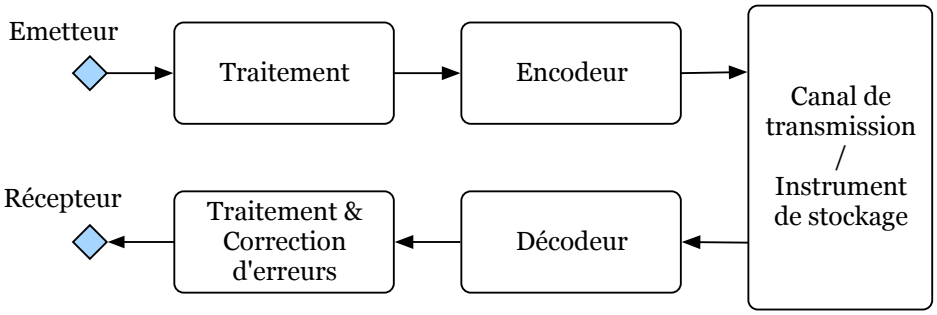


FIG. 1.2 – Synoptique de transmission de l'information vidéo

Les communications vidéo modernes utilisent la vidéo numérique qui est capturée par le biais d'une caméra ou synthétisée à partir d'outils appropriés tels que les logiciels d'animation. Le synoptique de la figure 1.2 représente la chaîne de transmission d'un flux vidéo. Dans une phase de pré-traitement optionnelle, l'émetteur peut choisir de traiter la vidéo en utilisant des techniques de perfectionnement (débruitage). Puis, l'encodeur comprime la séquence et la représente comme un flux binaire. Après la transmission du signal sur le canal de communication, le décodeur décomprime la vidéo qui peut être visualisée après un traitement éventuel des données (lissage des artefacts locaux).

La norme H.264/AVC définit la syntaxe et la sémantique du flux binaire encodé mais spécifie également les traitements que le décodeur doit réaliser pour décomprimer les données. En revanche, la norme ne définit pas comment encoder les données permettant ainsi aux entreprises de rentrer en concurrence dans des domaines tels que le coût, l'efficacité de codage, la robustesse du flux ainsi que les capacités matérielles.

Pour réaliser une transmission efficace dans des environnements divers, l'efficacité du codage ne représente pas la seule contrainte. En effet, l'encapsulation des données vidéo dans les protocoles et les architectures réseaux correspond à une étape majeure dans le développement

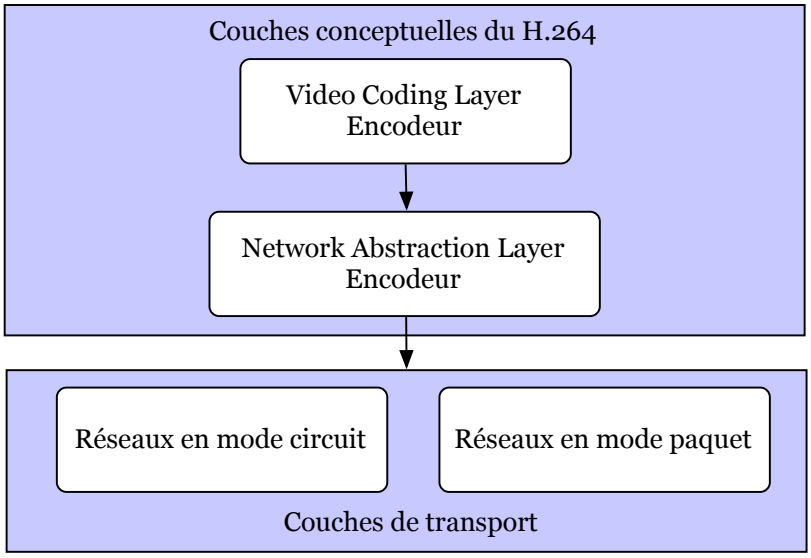


FIG. 1.3 – Les deux couches normalisées du H.264/AVC

d'un standard. L'adaptation du flux vidéo dans différents systèmes de transmission fut typiquement abordée dans les spécifications des normes MPEG précédentes, mais aussi dans H.320 et H.324. Cependant, seule une étroite intégration entre les protocoles réseaux et la vidéo codée peut apporter de bonnes performances. C'est pourquoi, H.264/AVC repose sur deux couches conceptuelles qui sont représentées sur la figure 1.3. La couche VCL (*Video Coding Layer*) définit une représentation efficace du flux vidéo tandis que la couche NAL (*Network Abstraction Layer*) convertit la structure VCL dans un format approprié aux systèmes de transmission actuels.

1.3.1 Généralités

Acquisition des images

Une image numérisée est représentée par une matrice de pixels (éléments de base) à deux dimensions, résultant d'un échantillonnage spatial de l'image se formant sur le capteur optique. Pour une image en noir et blanc, chaque cellule photoélectrique du capteur représente un pixel et fournit un signal électrique proportionnel à la quantité de lumière qu'elle reçoit. Pour une image en couleur, un pixel est caractérisé par trois cellules photoélectriques superposées, chacune recouverte d'un filtre coloré différent (rouge, vert et bleu). Chaque cellule permet de transformer la quantité de lumière colorée reçue en signal électrique. Un convertisseur analogique/numérique se charge ensuite de convertir ce signal en données binaires. Ainsi, chaque pixel d'une image en couleur est associé à trois composantes quantifiées RGB, où R représente l'intensité lumineuse de la composante rouge, G celle du vert et B celle du bleu. Chaque composante est généralement codée sur 8 bits et chaque pixel sur 24 bits.

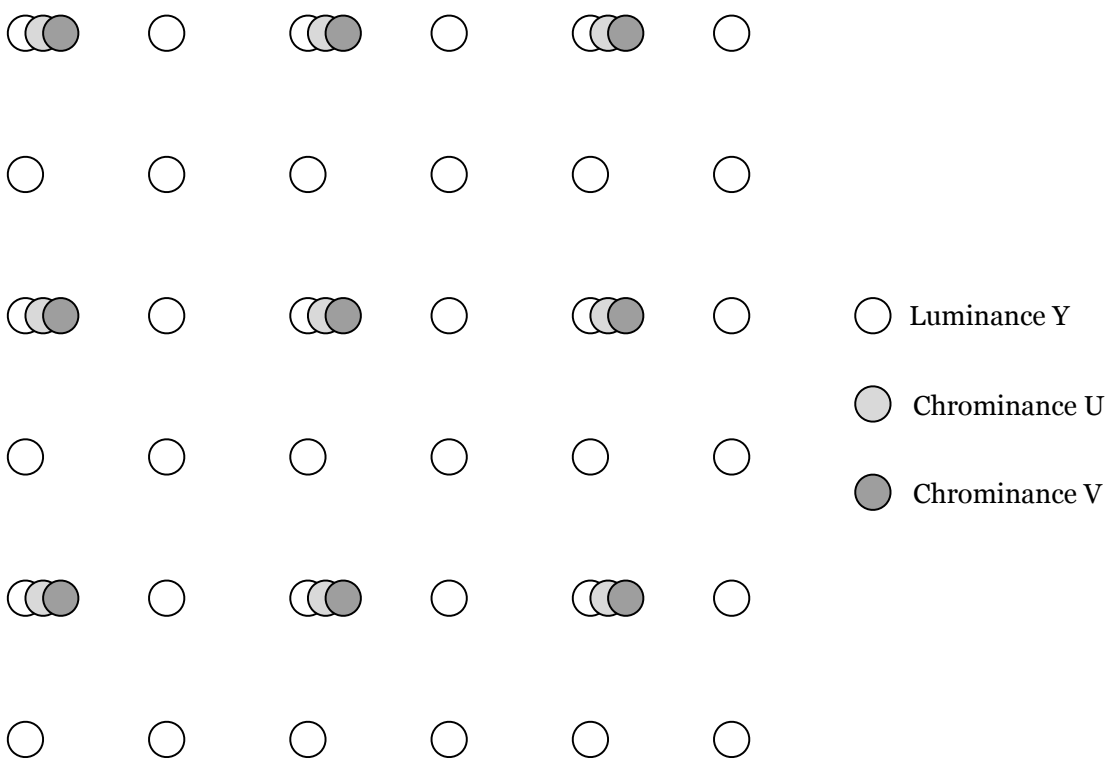


FIG. 1.4 – Représentation des composantes YUV pour le format 4 : 2 : 0

Toutefois, pour faciliter le stockage, il est plus commode de représenter les pixels de l'image au format YUV. Y est la luminance qui correspond à l'information de luminosité (niveau de gris). U et V définissent l'information liée à la couleur (chrominances bleu et rouge respectivement). Les trois composantes Y,U et V sont obtenues par combinaisons linéaires entre les composantes R,G et B. La vision humaine étant plus sensible à la luminance qu'à la chrominance, une première technique de compression consiste à sous-échantillonner les composantes de chrominance d'un facteur 2 dans les directions verticales et horizontales (format 4 : 2 : 0). Ce format populaire est illustré sur la figure 1.4. Avec ce genre de représentation, une image en couleur peut être décomposée en trois matrices bidimensionnelles : une matrice en pleine résolution contenant les pixels de luminance et deux matrices en quart de résolution contenant les pixels de chrominance.

Une vidéo correspond à une succession d'images fixes, rafraîchies à une cadence suffisamment importante pour donner au cerveau une sensation de mouvement (effet *phi*). Cette illusion de continuité est liée au système visuel humain qui comble automatiquement l'absence de transition entre les images. Une bonne impression de fluidité est obtenue à partir de 20 images par secondes. Le flux d'images est fourni par échantillonnage temporel du capteur optique.

Découpage hiérarchique des images

Contrairement aux codeurs en ondelettes, les codeurs hybrides ne réalisent pas directement un traitement de l'image complète. L'élément de base du traitement est le macrobloc. Un macrobloc représente une zone de 16×16 pixels pour la luminance et de 8×8 pixels pour la chrominance avec le format $4 : 2 : 0$. Pour comprimer l'image, le codeur décompose l'image en macroblocs et les encode successivement.

Certaines opérations de codage nécessitent de travailler avec une résolution inférieure à celle du macrobloc. La portion élémentaire d'un macrobloc est appelé un bloc. Un bloc correspond à une zone de 4×4 pixels. Par conséquent, un macrobloc de luminance est composé de 16 blocs et un macrobloc de chrominance au format $4 : 2 : 0$ est divisé en 4 blocs. La figure 1.5 représente la subdivision d'une image de luminance en macroblocs et en blocs.

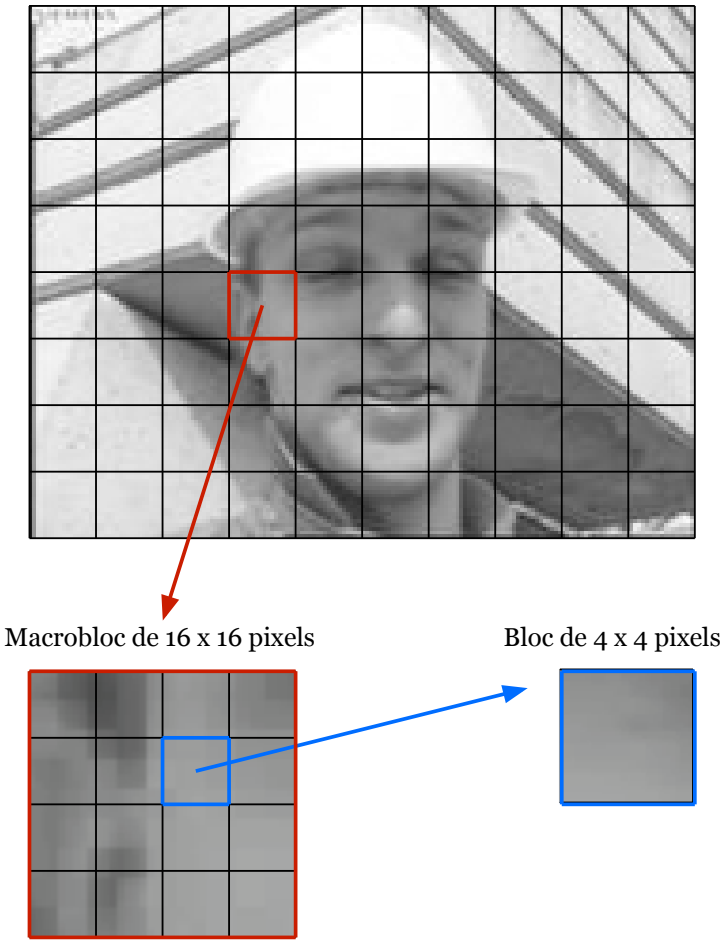


FIG. 1.5 – Subdivision d'une image en macroblocs et en blocs

Evaluation de la qualité

L'évaluation de la qualité d'une séquence vidéo est un problème complexe et souvent imprécis. Actuellement, cette évaluation peut-être réalisée par une analyse subjective ou objective. En règle générale, cette mesure permet d'évaluer l'impact visuel de la perte d'information (lors de la compression) ou des erreurs (résultant d'une transmission détériorée).

Dans la méthode subjective, un groupe d'individus est invité à juger la qualité des séquences vidéo. Les observateurs visualisent successivement deux séquences (originale puis traitée) et sont chargés de noter la qualité de la vidéo traitée en la comparant avec la vidéo originale. Toutes les procédures de test sont définies dans la norme ITU-R 500 [15]. Cette méthode d'analyse permet d'évaluer la qualité d'une vidéo à partir de nombreux critères, qui ne pourraient pas être considérés par un algorithme.

Dans de nombreuses situations, il est cependant préférable de pouvoir estimer la qualité des séquences vidéo automatiquement (sans faire appel à un jury). Ces méthodes caractérisent les techniques d'évaluation objectives. Certains programmes informatiques intègrent des traitements très simples, d'autres des algorithmes beaucoup plus complexes basés sur une analyse multi-critères. Dans le domaine de la compression vidéo, les contraintes temporelles sont fortes (applications temps-réel) et les méthodes traditionnelles consistent à mesurer le PSNR (*Peak Signal to Noise Ratio*) ou le MSE (*Mean Square Error*). Le PSNR correspond au rapport signal à bruit crête-à-crête et le MSE équivaut à l'erreur quadratique moyenne.

Ces deux critères permettent de mesurer la distorsion entre une image reconstruite I_r et une image de référence I_o (originale). En considérant que les images ont une taille de $L \times H$ pixels, le MSE est défini comme

$$MSE = \frac{1}{L \cdot H} \sum_{i=1}^{H} \sum_{j=1}^{L} \left[I_o(i,j) - I_r(i,j) \right]^2 , \tag{1.1}$$

et le PSNR est donné par

$$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right) , \tag{1.2}$$

lorsque les composantes de chaque pixel sont codées sur 8 bits.

En moyenne, les images reconstruites de qualité acceptable ont un PSNR supérieur à 30 dB.

1.3.2 La couche de codage vidéo (VCL)

Schéma général

La norme H.264/AVC représente actuellement l'aboutissement des techniques de compression hybrides et correspond à notre référence de codage source durant cette thèse. Le

schéma bloc généralisé d'un tel encodeur est illustré sur la figure 1.6.

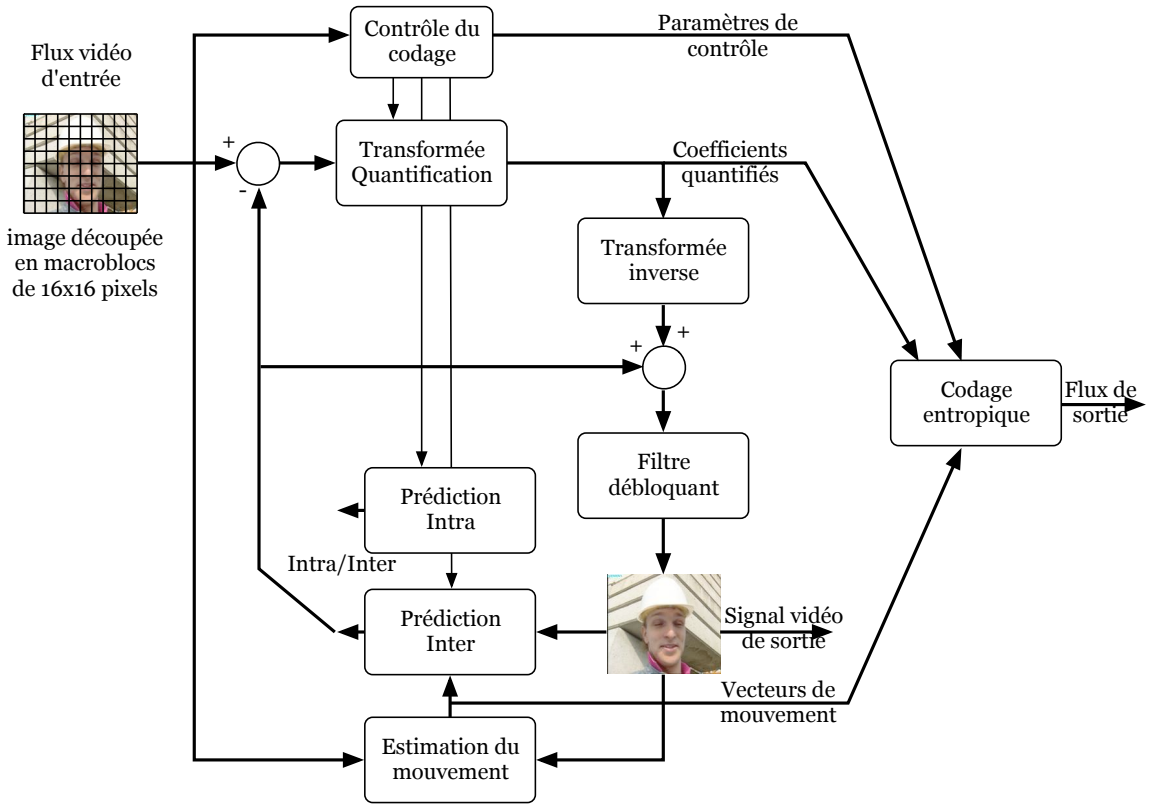


FIG. 1.6 – Structure de codage principale du H.264/AVC

L'image d'entrée est divisée en macroblocs. Chaque macrobloc est composé de trois composantes : Y, U et V. Du fait que la vision humaine est moins sensible à la chrominance qu'à la luminance, les macroblocs de chrominance sont sous-échantillonés d'un facteur 2 dans les directions horizontales et verticales. Par conséquent, chaque portion élémentaire de l'image est composée d'un macrobloc de luminance de 16×16 pixels et de deux macroblocs de chrominance de 8×8 pixels.

Chaque macrobloc est prédit en mode Intra ou Inter. Ces deux outils caractérisent l'étage de prédiction du codeur. Le mode Intra exploite les redondances spatiales des images, il permet de construire une estimation d'un macrobloc en utilisant exclusivement les informations contenues dans l'image courante. Le mode Inter tire parti des redondances temporelles entre les images, il permet de prédire le macrobloc courant en utilisant les informations contenues dans des images de référence, qui ont déjà été encodées, décodées puis stockées dans une mémoire. Ce principe de compensation en mouvement repose sur l'estimation d'un vecteur de déplacement associé à chaque bloc. Ce vecteur caractérise la position du bloc le plus vraisemblable dans l'image de référence. Il est évident que la prédiction Inter est beaucoup plus efficace que la prédiction Intra car les redondances temporelles représentent une forte proportion de l'énergie du signal. Le mode Inter est donc utilisé en priorité dans les codeurs vidéo,

excepté dans la situation où la mémoire ne contient aucune image de référence (première image d'un film par exemple).

L'erreur de prédiction, qui correspond à la différence entre le bloc original et le bloc prédit, est ensuite transformée, quantifiée puis encodée par le biais d'un codage entropique. Les informations de contrôle ainsi que les vecteurs de mouvement sont également encodés avant d'être transmis. De manière à reconstruire les mêmes images à l'encodeur et au décodeur, les coefficients quantifiés de chaque bloc sont inversés et ajoutés au signal de prédiction. Cette opération permet de reconstruire chaque macrobloc encodé, qui pourra ensuite servir de référence pour la prédiction des autres macroblocs. Afin de réduire les artefacts entre les blocs, un filtre débloquant a été intégré dans la boucle de compression. Ce dernier permet de lisser les informations visuelles avant de les stocker dans la mémoire de référence.

La structure de traitement du décodeur est plus simple que celle de l'encodeur. Il commence par décoder les différents types d'information : paramètres de contrôle, vecteurs de mouvement et coefficients quantifiés. Les macroblocs sont ensuite successivement prédits en utilisant le mode approprié (Intra ou Inter). En parallèle, les résidus de prédiction sont reconstruits grâce aux coefficients quantifiés, puis ajoutés au signal de prédiction. Suite à l'opération de filtrage, le macrobloc est complètement décodé et peut être stocké en mémoire pour les prédictions futures.

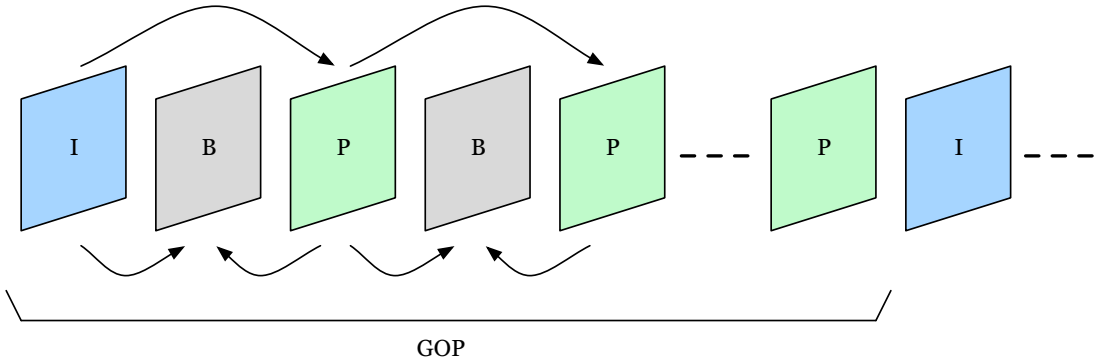


Fig. 1.7 – Structure d'un GOP et dépendances entre images

Cinq types d'images sont supportés par la norme H.264/AVC et sont nommés : I, P, B, SI et SP. Dans une image I, tous les macroblocs sont prédits en utilisant le mode Intra. De manière idéale, une image I devrait intervenir lors d'un changement de scène, c'est-à-dire lorsque les redondances temporelles entre les images sont faibles. Dans la pratique, les programmes de détection de changement de scène sont délicats à implémenter et les codeurs insèrent généralement une image I à intervalles réguliers (une image I apparaît communément toutes les secondes). Dans une image P, chaque macrobloc est prédit en utilisant soit le mode Intra, soit le mode Inter. Lorsque le mode Inter est activé, chaque macrobloc est associé à une seule image de référence. Contrairement aux images P, les macroblocs d'une image B construits avec le mode Inter peuvent s'appuyer sur deux images de référence. Les images SI et SP sont des représentations spécifiques qui sont exploitées pour réaliser une commutation

efficace entre deux flux différents. Ces formats d'images sont rarement utilisés en pratique. Les différents types d'images sont regroupés pour former des séquences (*Group Of Pictures* ou GOP). Un GOP débute par une image I et contient ensuite une succession d'images P et B. La structure classique d'un GOP est illustrée sur la figure 1.7. Généralement, la première image d'un GOP vide la mémoire de référence (*Instantaneous Decoding Refresh* ou IDR). Par conséquent, les GOPs sont indépendants entre eux. Cette technique permet au décodeur de se resynchroniser sur le flux dans le cas d'une transmission avec pertes.

Dans la suite, nous présentons en détail le principe de fonctionnement des différents outils intervenant dans le codeur.

La prédiction Intra

Ce mode de prédiction [16] consiste à estimer les échantillons d'un macrobloc en utilisant les informations contenues dans les blocs contigus appartenant au passé spatial de l'image courante. Ces blocs de référence doivent déjà avoir été encodés puis décodés par le codeur (de manière à retrouver des résultats identiques au codeur et au décodeur). La norme H.264/AVC propose deux types de traitement Intra pour prédire le signal de luminance.

Le premier mode est appelé Intra4x4 et le second Intra16x16. Dans le type Intra4x4, le macrobloc est divisé en seize blocs de 4×4 pixels et chaque bloc est encodé individuellement. Neuf modèles de prédiction sont fournis par la norme et l'objectif du codeur est de sélectionner le mode le plus adapté au bloc courant. Ces neuf modes sont représentés sur la figure 1.8. Nous pouvons constater que huit de ces modes caractérisent des modèles de prédiction directionnels. Chaque valeur prédite du bloc courant est déterminée à partir d'une combinaison linéaire entre les pixels situés dans les blocs contigus. Ces combinaisons linéaires sont définies dans la norme en fonction d'une direction spécifique. Seul le mode DC n'est pas associé à une direction et consiste juste à calculer la moyenne des pixels contenus dans les deux blocs voisins.

Lorsque le type Intra16x16 est utilisé, seul un mode de prédiction est appliqué pour encoder l'ensemble du macrobloc. Quatre modèles de prédiction sont définis dans la norme et l'objectif du codeur est de choisir le mode le plus adapté au macrobloc courant. Ces quatre modes sont représentés sur la figure 1.9. Nous pouvons noter que trois de ces modes sont associés à des directions spécifiques (verticale, horizontale et plane). Le dernier mode (DC) correspond à un calcul de moyenne. La prédiction Intra16x16 s'avère très efficace pour coder les régions où la luminance varie doucement car l'estimation s'applique au macrobloc complet.

Pour le codage des macroblocs de chrominance U et V, seule la prédiction Intra8x8 est utilisée. Ce schéma est suffisant car les variations des signaux de chrominance sont très faibles. Le mode Intra8x8 permet d'estimer les 8×8 pixels d'un macrobloc de chrominance en proposant quatre modèles de prédiction : DC, vertical, horizontal et plan. Les types Intra8x8 et Intra16x16 sont donc identiques à un facteur d'échelle près.

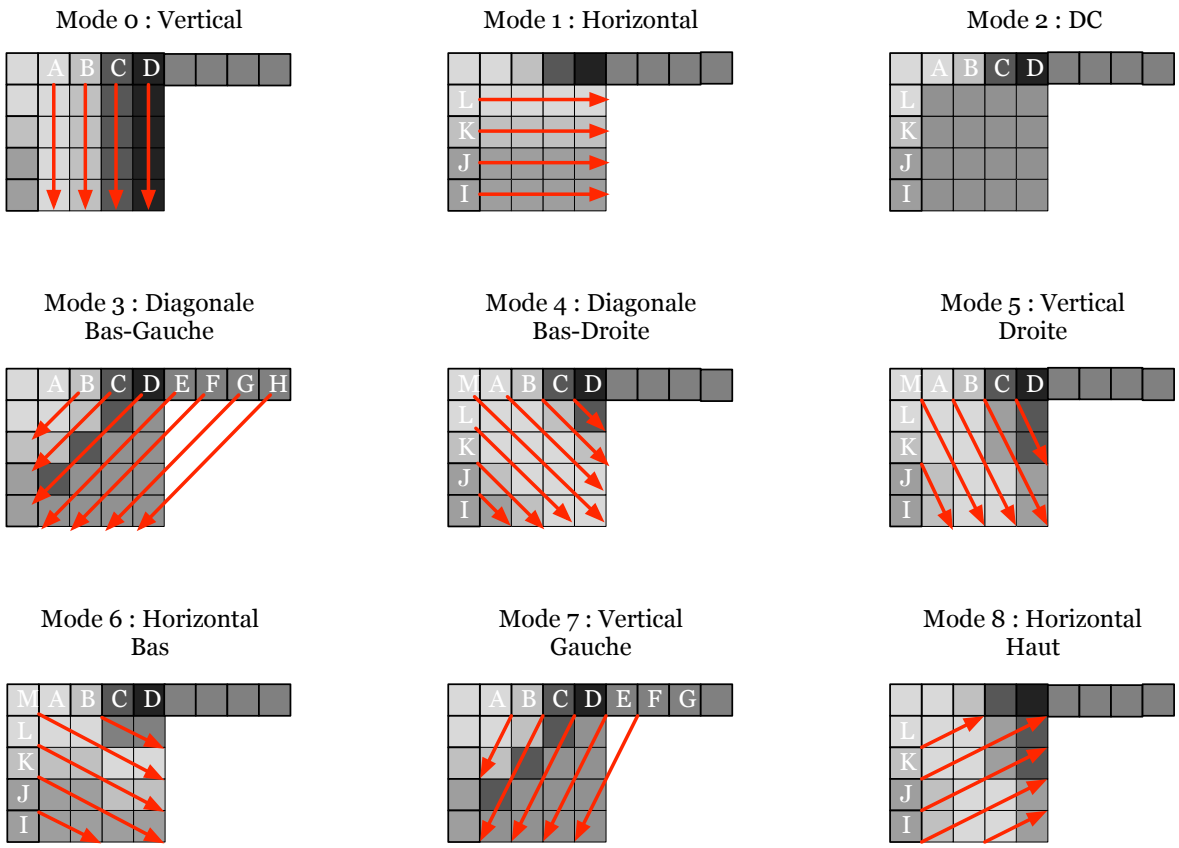


FIG. 1.8 – Les neuf modes de la prédiction Intra4x4

La prédiction Inter

Dans le cas de la prédiction Inter [16], les macroblocs d'une trame sont prédits à partir des échantillons d'une image de référence précédemment encodée (puis décodée). Afin de réaliser précisément cette opération, chaque macrobloc peut être divisé en partitions de tailles variables. Ainsi, une partition de luminance peut être composé de 16×16, 16×8, 8×16 ou 8×8 échantillons. Un sous-macrobloc de 8×8 pixels peut être à nouveau redécoupée en partitions de taille 8×4, 4×8 ou 4×4. Cette décomposition pyramidale permet d'isoler les différents objets composant une image et de s'adapter à leurs caractéristiques (sens de déplacement, vitesse). Un vecteur de mouvement est associé à chaque partition d'un macrobloc. Ce vecteur de déplacement spécifie l'écart spatial entre la partition courante de l'image actuelle et sa meilleure représentation dans l'image de référence. Les partitions d'un macrobloc et d'un sous-macrobloc sont illustrées sur la figure 1.10.

Dans les anciennes normes telles que MPEG-4 ou H.263, seuls des partitions de 16×16 et 8×8 pixels étaient supportés par le codage Inter. Dans H.264/AVC, les combinaisons de découpage sont beaucoup plus nombreuses et permettent d'obtenir des formes arbitraires. Par ailleurs, il est maintenant possible de se référencer à plusieurs images antérieures. Dans

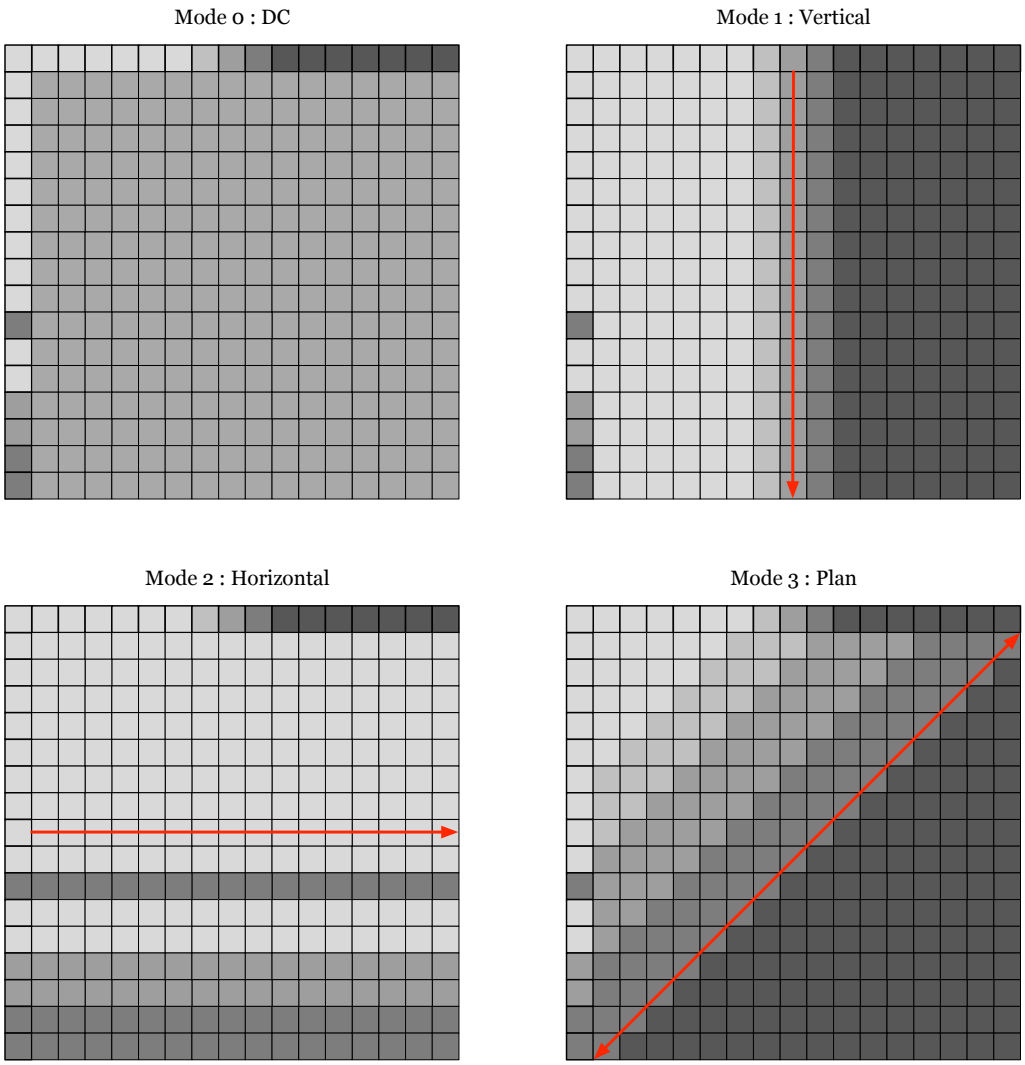


FIG. 1.9 – Les quatre modes de la prédiction Intra16x16

ce but, un paramètre additionnel d'indexation des images doit être transmis avec le vecteur de déplacement de chaque partition. Cette technique correspond à la prédiction compensée en mouvement avec des images de référence multiples. Elle est illustrée sur la figure 1.11.

La précision des vecteurs de mouvement peut atteindre le quart de pixel. Un tel déplacement (résolution fractionnelle) peut pointer sur des positions qui sont spatialement situées entre les échantillons du signal image. Le signal de l'image de référence doit donc être interpolé entre les pixels. Dans la norme H.264/AVC, le sur-échantillonnage d'un facteur 2 du signal de luminance est généré en appliquant un filtre RIF uni-dimensionnel d'ordre 6 sur les pixels de l'image. Les éléments correspondant au quart de pixel sont obtenus en moyennant les échantillons intermédiaires du signal. Le signal de chrominance par le calcul d'une simple moyenne quelque soit le niveau de résolution.

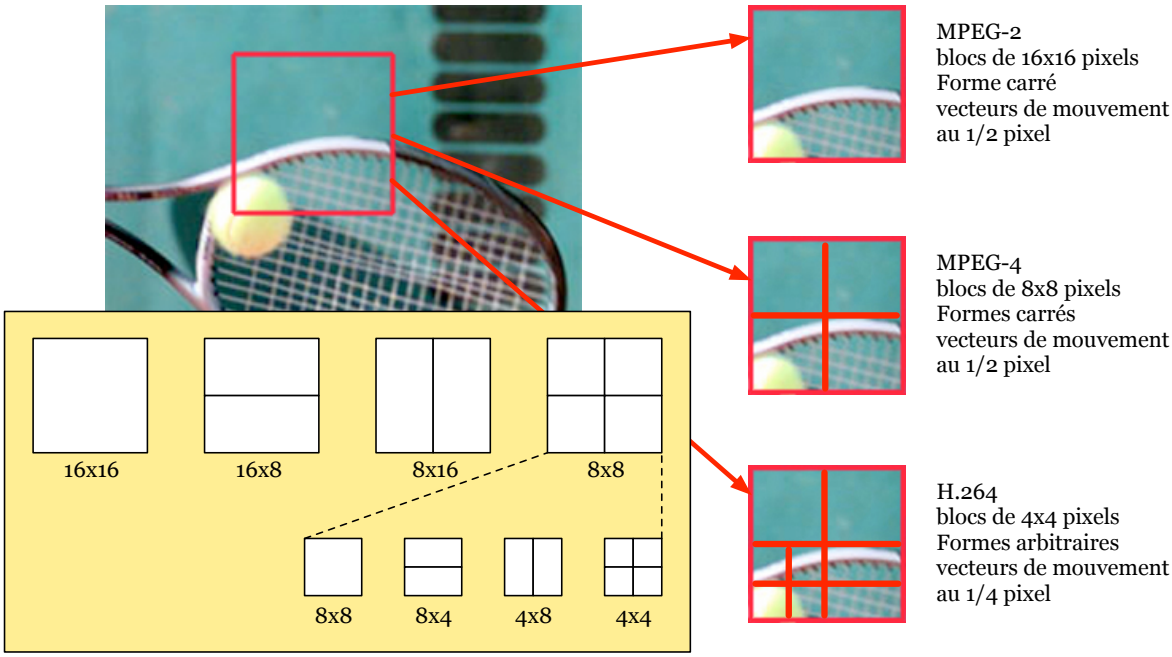


FIG. 1.10 – Les modes de prédiction Inter

Le concept classique des images de type B avait déjà été abordé dans les anciens standards de compression vidéo. Dans cette approche, chaque partition est construite en utilisant deux images de référence (généralement les images $n-1$ et $n+1$). Le codeur recherche les ensembles de référence les plus adaptés à la partition courante. La prédiction consiste ensuite à calculer la moyenne de chaque pixel. Le H.264/AVC généralise cette notion et permet de fixer des paramètres de pondération arbitraires.

Le codage par transformée

Dans le même contexte que les normes précédentes, la phase de transformation-quantification est appliquée dans le but de coder le signal d'erreur de prédiction. La tâche de la transformée consiste à réduire les redondances spatiales du signal d'erreur. Tous les anciens standards tels que le MPEG-1 et MPEG-2 appliquaient une DCT de taille 8×8 sur chaque bloc de l'image. En revanche, le H.264/AVC dispose d'une transformée basée sur des entiers [17]. La matrice de transformation est généralement composée de 4×4 éléments, mais peut être réduite à 2×2 éléments pour le codage de certaines informations de chrominance. La diminution de la taille de la fenêtre d'analyse permet à l'encodeur de mieux adapter le codage de l'erreur de prédiction aux frontières des objets mouvants. En effet, la taille du bloc est similaire aux dimensions de la plus petite zone d'analyse de l'estimation Inter ou Intra (4×4 pixels) et la transformée s'ajuste donc mieux aux erreurs de prédiction locales.

Il existe trois types différents de transformées. Le premier est appliqué à tous les échantil-

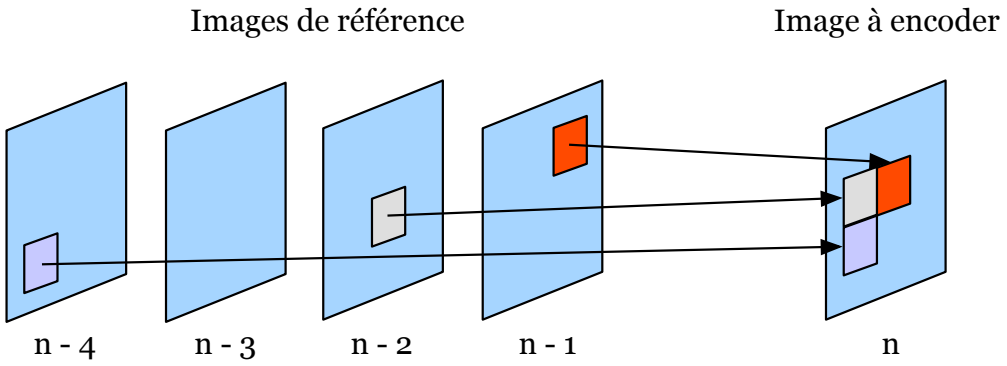


FIG. 1.11 – Prédiction compensée en mouvement avec images de référence multiples

lons d'erreurs à la fois pour le signal de luminance Y mais aussi pour les composantes de chrominance U et V, quelque soit le mode de prédiction utilisé (Inter ou Intra). Cette matrice de transformation H_1 est composée de 4×4 éléments. Sa structure est exposée en (1.3).

$$\mathbf{H}_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}. \tag{1.3}$$

Si le macrobloc est prédit en utilisant le mode Intra16x16, la seconde transformée est appliquée en plus de la première. Cette dernière convertit les seize coefficients DC des blocs transformés d'un macrobloc de luminance. Elle correspond à une transformée de Hadamard dont la taille s'élève à 4×4 composantes. Une représentation de la matrice est donnée en (1.4).

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}. \tag{1.4}$$

Le troisième type se rapporte aussi à une transformée de Hadamard mais de taille 2×2. Elle est utilisée pour le codage des quatre coefficients DC contenus dans un macrobloc de chrominance. Sa matrice est représentée en (1.5).

$$\mathbf{H}_3 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{1.5}$$

L'opération de transformation dans H.264/AVC se traduit par l'équation

$$Y = H_i \cdot X \cdot H_i^T, \tag{1.6}$$

où Y est la matrice transformée, X le signal d'entrée et H_i peut représenter H_1 (1.3), H_2 (1.4) ou H_3 (1.5).

L'ordre de transmission de tous les coefficients est représenté sur la figure 1.12. Si le macrobloc est prédit en utilisant le mode Intra16x16, le bloc muni de l'indexe -1 est diffusé en premier. Ce paquet contient les coefficients DC de tous les blocs de luminance. Tous les ensembles indicés de 0 à 25 sont ensuite transmis. Les éléments numérotés de 0 à 15 correspondent à tous les coefficients AC de la luminance. Les blocs 16 et 17 représentent les composantes DC de chaque signal de chrominance. Enfin, les valeurs indexées de 18 à 25 se rapportent aux coefficients AC de la chrominance.

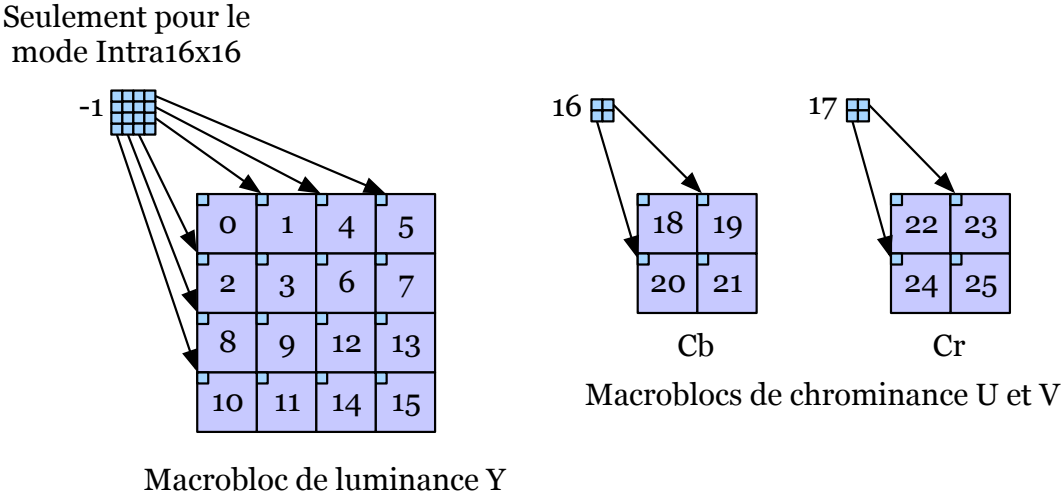


FIG. 1.12 – Ordre de transmission de tous les coefficients d'un macrobloc

En comparaison avec la DCT, les matrices de transformation du H.264/AVC sont composées seulement de nombres entiers dans un intervalle compris entre -2 et $+2$. Ce principe permet de calculer la transformée et son inverse sur seize bits en utilisant seulement des opérations de décalages, des additions et des soustractions. Dans le cas d'une projection de Hadamard, seules l'addition et la soustraction sont nécessaires. De plus, les disparités liées aux approximations du calcul flottant sont complètement évitées grâce à l'utilisation exclusive d'opérations sur des entiers.

Tous les coefficients sont ensuite quantifiés par le biais d'un quantificateur scalaire. La taille du pas de quantification est choisie par un paramètre QP qui supporte cinquante deux valeurs possibles. La taille du pas double lorsque la variable QP est incrémentée de 6. Une augmentation de QP de 1 entraîne un accroissement du débit des données d'environ $12,5$ %.

Le codage entropique

La procédure d'encodage entropique introduit les propriétés sémantiques et syntaxiques dans le flux vidéo comprimé. Dans le chapitre 3, nous exploitons ces propriétés pour améliorer la qualité du décodage vidéo au niveau du récepteur. Il est donc essentiel de définir précisément

les opérations intervenant durant cette phase de codage. C'est pourquoi, cette partie est plus détaillée que les autres.

Le H.264/AVC propose deux méthodes alternatives de codage entropique : une technique de faible complexité basée sur l'usage de contextes adaptatifs contenant des mots de code VLC, nommée CAVLC (*Context-based Adaptive Variable Length Coding*) [18], et un algorithme plus coûteux fondé sur un codage arithmétique reposant sur des tables évolutives, le CABAC (*Context-based Adaptive Binary Arithmetic Coding*) [19]. Les deux méthodes représentent des améliorations majeures en terme d'efficacité de compression en comparaison avec les techniques de codage statistique traditionnelles. Dans les anciens standards, l'encodage de chaque élément de syntaxe était basé sur des tables VLC fixes (une distribution de probabilité était associée à chaque élément). Cependant, des études pratiques ont rapidement démontré que les signaux étaient rarement stationnaires et que l'utilisation de tables adaptatives (contextuelles) était plus efficace pour comprimer les données. Des modèles contextuels ont donc été intégrés dans le processus d'encodage entropique.

Durant cette thèse, nous avons exclusivement utilisé la méthode CAVLC et nous avons délaissé le codeur arithmétique. Ce choix se justifie pour deux raisons : il fournit un codage efficace de faible complexité et il est plus adapté aux applications liées aux télécommunications. De plus, à la différence du CABAC, le CAVLC est inclu dans tous les profils définis par la norme. Nous ne présenterons donc pas le codeur arithmétique dans ce rapport.

Dans le codage CAVLC, deux techniques de compression sont utilisées. La première, basée sur un codage Exponential-Golomb (noté Exp-Golomb dans la suite) [20], se charge d'encoder tous les paramètres de codage (type de macrobloc, pas de quantification, vecteurs de mouvement, etc) à l'exception des résidus de prédiction. Ces résidus sont encodés par la deuxième méthode, plus compliquée, mais permettant de comprimer les données de manière adaptative.

Le codage Exp-Golomb Dans ce type de codage, tous les symboles d'entrée sont représentés par des entiers. Ils sont ensuite convertis en mots de code VLC par l'intermédiaire de la table de correspondance exposée sur la figure 1.13. Certains symboles peuvent correspondre à des valeurs positives ou nulles, d'autres à des nombres signés. A titre d'exemple, le mode de prédiction Intra16x16 d'un macrobloc est défini par une valeur entière allant de 0 à 3 (pour les quatre modes proposés par la norme). A l'inverse, les vecteurs de mouvement peuvent être des valeurs entières positives, négatives ou nulles.

Nous pouvons constater que les codes Exp-Golomb sont composés d'un préfixe et d'un suffixe. Le préfixe représente la première portion du mot de code, il contient un ensemble de zéros et se termine par 1. Le nombre de zéros contenus dans le préfixe spécifie la taille du suffixe. Lors de la réception, le décodeur récupère le préfixe et en déduit la taille binaire du suffixe.

La table représentée sur la figure 1.13 nous montre également que les mots de code les plus courts sont attribués aux symboles les plus faibles (en valeur absolue). La distribution de

Valeur positive	Valeur signée	Mot de code
0	0	1
1	+1	010
2	−1	011
3	+2	00100
4	−2	00101
5	+3	00110
6	−3	00111
7	+4	0001000
8	−4	0001001
9	+5	0001010
10	−5	0001011
...	...	...

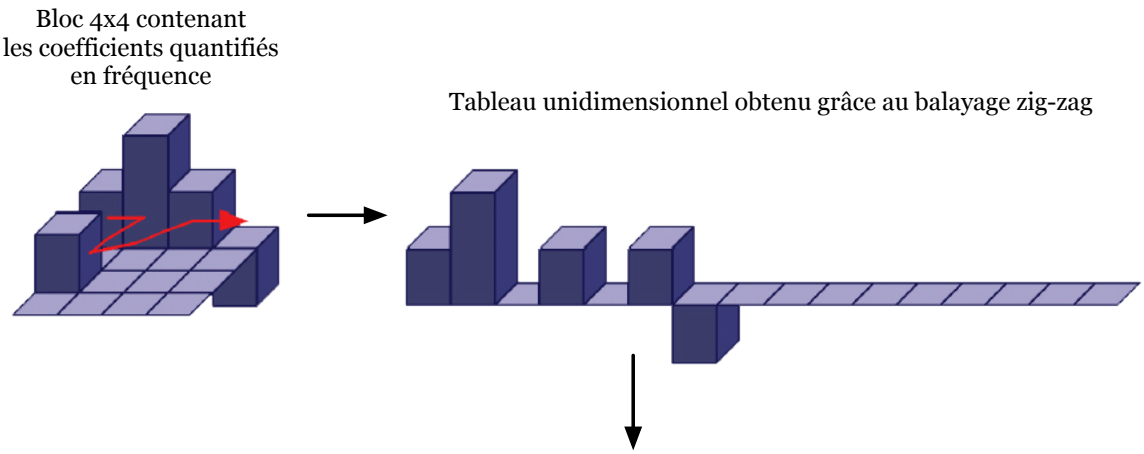
FIG. 1.13 – Table de correspondance du codage Exp-Golomb

probabilité d'un symbole doit donc être concentrée autour de 0. Cependant, tous les symboles ne respectent pas directement cette distribution. C'est le cas des vecteurs de mouvement ou du pas de quantification par exemple. L'encodeur doit alors procéder à une étape de prédiction ou de translation. Concernant les vecteurs de mouvement, ces derniers sont prédits dans une phase préliminaire en utilisant les vecteurs de mouvement des blocs situés au-dessus et à gauche du bloc courant. Seul l'offset résiduel (résultant de la différence entre le vecteur exact et sa prédiction) est encodé en Exp-Golomb, puis transmis. Lors de l'opération de codage du pas de quantification moyen d'une image, les valeurs possibles varient entre 0 et 51. L'encodeur soustrait donc la valeur choisie de 26 pour ramener l'ensemble des valeurs dans l'intervalle $[-26, +25]$. La valeur résultante est ensuite codée à partir de la table.

Le codage des résidus Dans le codage CAVLC, les résidus de prédiction (transformés et quantifiés) sont encodés de manière indépendante en suivant une procédure spécifique. Cette méthode sophistiquée est résumée sur la figure 1.14. Dans cette partie, nous expliquons brièvement le principe d'encodage d'un bloc 4×4. Une description plus détaillée de la méthode est présentée dans l'annexe A.

Les seize coefficients du bloc transformé sont d'abord répartis dans un tableau unidimensionnel en respectant un schéma de balayage spécifique (*zig-zag scan*). Cette technique permet notamment de regrouper les coefficients quantifiés faibles ou nuls à la fin du tableau. Après cette opération, les zéros sont statistiquement rassemblés dans les hautes fréquences et une occurrence prédominante de niveaux égaux à $+/-1$ (notés T1s) sont situés à la fin du tableau.

Le nombre de coefficients non-nuls (*TotalCoeffs*) et le nombre de T1s (*TrailingOnes*) sont encodés en premier lieu. Ces deux paramètres sont combinés dans un simple mot de code (*CoeffToken*), extrait d'une des quatre tables VLC définies sur la figure A.1. La sélection de la table VLC dépend du nombre de coefficients non-nuls contenus dans les blocs voisins.



	Codage CAVLC
Préambule	TotalCoeffs = 5 TrailingOnes = 3
Eléments de syntaxe parcourus dans le sens inverse	Signe T1s : -1, +1, +1 Levels : +2, +1 TotalZeros = 2 RunBefores : 0, 1, 1

FIG. 1.14 – Codage des résidus de prédiction d'un bloc 4×4 avec la méthode CAVLC

Dans une deuxième étape, le signe et l'amplitude de chaque coefficient non-nul sont encodés en parcourant le tableau dans le sens inverse (en partant des hautes fréquences pour aboutir aux basses fréquences). Pour le codage des T1s, seul le signe est nécessaire. Il est représenté par un simple bit. Concernant les autres coefficients non-nuls (*Levels*), la procédure est plus complexe. Cette technique considère que l'amplitude des coefficients augmente lorsque la fréquence diminue. De manière simplifiée, la table VLC, utilisée pour encoder l'amplitude et le signe d'un coefficient, est construite en fonction du coefficient précédent.

Le nombre de coefficients nuls précédant le dernier coefficient non-nul (*TotalZeros*) est encodé dans une troisième étape. Son encodage dépend des tables VLC illustrées sur la figure A.4. La table appropriée est choisie en fonction de la valeur de *TotalCoeffs*. Le paramètre *TotalZeros* est suivi des *RunBefores* qui indiquent le nombre de zéros consécutifs situés entre chaque coefficient non-nul. Chaque *RunBefore* est codé séparément en utilisant les tables de la figure A.6. La table adaptée dépend du nombre de coefficients nuls restant à encoder.

Dans des conditions typiques d'encodage, ce principe permet d'obtenir des réductions

de débit de 2 à 7 % par rapport aux schémas conventionnels basés sur de simples codes Exp-Golomb.

Le filtre débloquant

Une caractéristique particulière du codage par blocs correspond à la production accidentelle d'artefacts entre des ensembles successifs. Le filtrage du bord des blocs représente un outil puissant qui permet de réduire considérablement la visibilité de ces disparités [21]. Dans le principe, le lissage peut être considéré comme un traitement final, se rapportant seulement aux images qui sont affichées. Une plus haute qualité visuelle peut encore être atteinte en intégrant le filtre dans la boucle d'encodage. En effet, toutes les trames de référence passées, utilisées pour la compensation en mouvement, seront des versions corrigées des images reconstruites. C'est pour cette raison, que le H.264/AVC incorpore cette technologie dans la boucle de traitement. Ce filtre est hautement adaptatif car il dépend de plusieurs éléments de syntaxe mais aussi des caractéristiques locales de l'image. Ces différentes contraintes permettent de contrôler la souplesse du traitement de filtrage.

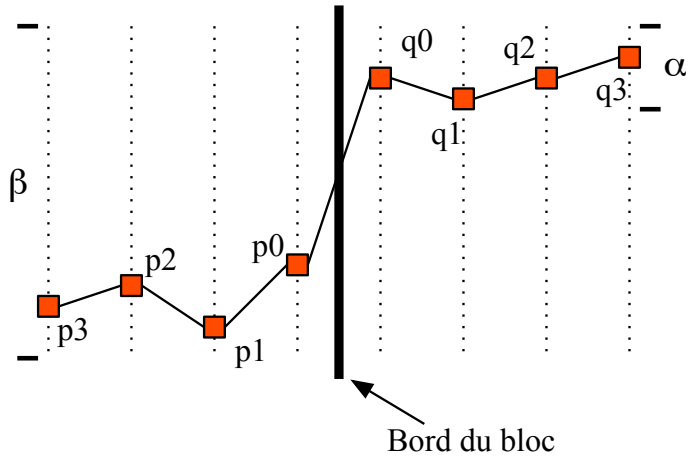


FIG. 1.15 – Principe du filtrage des blocs

La figure 1.15 illustre le principe de l'opération en utilisant une représentation d'un bord à une dimension. Avant de réaliser le traitement, des conditions doivent être vérifiées. Ces contraintes dépendent de la valeur des échantillons et du paramètre de quantification QP. Ainsi, le filtrage de p_0 et q_0 intervient seulement si chacune des expressions suivantes est vérifiée :

1. $|p_0 - q_0| < \beta(QP)$
2. $|p_1 - p_0| < \alpha(QP)$
3. $|q_1 - q_0| < \alpha(QP)$

où le seuil $\alpha(QP)$ est considérablement plus faible que $\beta(QP)$.

L'idée de base repose sur une simple constatation. Si la différence entre le échantillons proches du bord d'un bloc est relativement importante, il est probable que ce phénomène corresponde à un artefact de bloc et doive donc être réduit. En revanche, en cas d'écart trop élevé, ne pouvant donc pas être expliqué par la brutalité de la quantification, le signal représente plutôt l'information propre de l'image source. Dans ce dernier cas, le lissage n'est pas appliqué au bloc.

FIG. 1.16 – Performance du filtre de lissage pour des séquences fortement comprimées

La figure 1.16 illustre les performances d'un tel filtre. Il est aisé de remarquer que les artefacts liés aux jonctions entre les blocs ont été considérablement réduits tandis que le contenu vidéo est resté inchangé. Par conséquent, la qualité subjective a été améliorée de façon significative. Ce traitement permet ainsi de réduire le débit de 5 à 10 % tout en conservant la même qualité visuelle.

1.3.3 La couche réseau (NAL)

Dans les parties précédentes, nous avons constaté que la couche VCL fournit les outils nécessaires pour comprimer une image sur un nombre restreint de bits. Les macroblocs d'une image sont encodés successivement et le flux généré par la couche VCL correspond donc à une succession de séquences binaires caractérisant chaque macrobloc encodé. Le format du flux encodé pour une image CIF (*Common Intermediate Format,* 352×288 *pixels*) est illustré sur la figure 1.17. Un en-tête (*Image header*) est inséré au début du flux et définit la valeur des paramètres généraux de l'image.

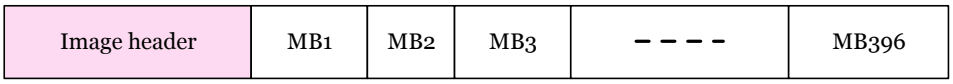


FIG. 1.17 – Format du flux encodé correspondant à une image CIF

En revanche, une vidéo comprimée ne se limite pas à une succession d'images encodées. D'autres informations sont nécessaires pour assurer la compatibilité entre l'encodeur et le décodeur. Par conséquent, la couche VCL génère des flux complémentaires qui contiennent les paramètres additionnels de la vidéo. Ces flux de données sont insérés entre les images lors du stockage ou de la transmission. A titre d'exemple, ces flux peuvent représenter les paramètres importants s'appliquant à une large séquence d'images (SPS ou *Sequence Parameter Set*) ou définissant les caractéristiques liées à l'affichage des images (SEI ou *Supplemental Enhancement Information*).

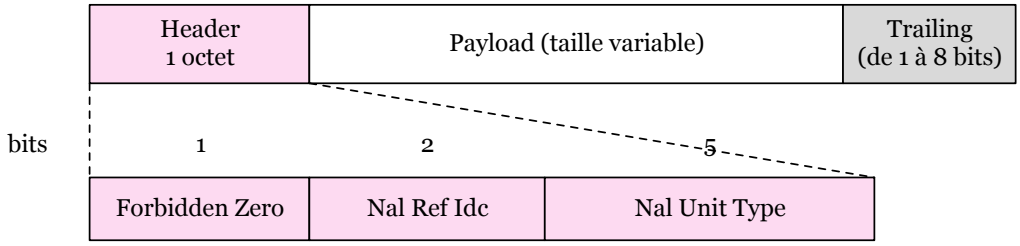


FIG. 1.18 – Format d'un paquet NAL

Contrairement à la couche VCL, la couche NAL convertit la structure VCL dans un format approprié aux systèmes de communication actuels. A un niveau général, la vidéo encodée est répartie dans des paquets NAL qui contiennent un nombre entier d'octets. Les informations utiles (flux VCL) sont encapsulées dans un champ de contenu (*payload*). Un en-tête (*header*) de 1 octet est ajouté au début de chaque paquet et précise le type de données transportées. Un champ de bourrage (*trailing bits*) est ajouté à la fin de chaque paquet et permet de ramener la taille du paquet à un nombre entier d'octets. Le format des paquets NAL est illustré sur la figure 1.18. Les champs d'information composant le paquet sont spécifiés ci-dessous :

- Le champ *Forbidden Zero* de 1 bit doit toujours être égal à 0.
- Les 2 bits du champ *Nal Ref Idc* spécifie le degré d'importance des informations contenues dans le paquet. Plus précisément, ce paramètre indique si l'image encodée du paquet correspond à une image de référence. Plus cette valeur est grande (tend vers 3), plus l'image a d'influence sur la qualité vidéo générale.
- Les 5 bits du champ *Nal Unit Type* spécifie la nature des données transportées dans le paquet. Comme nous l'avons énoncé plus haut, il existe différentes catégories d'information (image encodée, SEI, SPS, etc) et ce paramètre définit ce type.
- La *payload* contient les données utiles à transmettre : elle représente les informations fournies par la couche VCL. Ce champ peut contenir un nombre variable de bits car

chaque image encodée a une taille variable.
– Le champ *Trailing* représente des données de remplissage. Son bit de poids fort vaut 1 et les bits suivant sont fixés à 0. Ces bits permettent d'ajuster la taille du paquet pour obtenir un nombre entier d'octets.

Pour les circuits à commutation, la couche NAL délivre la vidéo codée comme un flux ordonné d'octets. Les paquets NAL sont envoyés successivement sur le canal et sont séparés par des marqueurs de synchronisation qui permettent au décodeur d'identifier chaque entité de données. Pour les transmissions en mode paquet (de type RTP/UDP/IP ou TCP/IP), les paquets NAL sont encapsulés dans des paquets RTP. Dans cette thèse, nous nous intéressons à ce dernier type de transport et le chapitre 2 décrit cette procédure en détail.

1.3.4 Outils optionnels de protection

La norme H.264/AVC inclut également de nombreuses options destinées à la rendre plus robuste aux erreurs lors d'une transmission sur un canal bruité. Dans cette partie, nous décrivons brièvement ces outils.

Le découpage des images

Lors de l'analyse du schéma de codage, nous avons indiqué que le codeur travaillait sur des images complètes. Il découpait l'image en macroblocs, les encodait successivement en commençant par la première rangée jusqu'à atteindre la dernière (*raster scan*) et finalement encapsulait les données encodées dans un paquet NAL. Dans certaines situations, il est cependant préférable de découper l'image complète en plusieurs régions (*slices*) contenant chacune un nombre entier de macroblocs. Un tel type de découpage est illustré sur la figure 1.19. Dans cet exemple, l'image de base est découpée en trois *slices*. Dans une utilisation basique, cette technique permet de répartir les données associées à chaque *slice* dans des paquets NAL séparés. Elle permet juste de répartir les informations d'une image sur plusieurs paquets et correspond donc à une étape de fragmentation préliminaire.

Lorsque le mode ASO (*Arbitrary Slice Ordering*) [14] est activé, le codeur vidéo encode les *slices* de manière indépendante. En somme, les macroblocs d'un *slice* ne dépendent pas des macroblocs appartenant à un autre *slice*. Les macroblocs sont traités successivement en respectant l'ordre traditionnel et chaque *slice* comprimé est associé à un paquet NAL. Au niveau du décodeur, un paquet perdu ne pourra pas détruire la totalité de l'image (seulement une partie des informations visuelles sera perdue). Par ailleurs, les données contenues dans les *slices* correctement reçus pourront servir à estimer les régions manquantes des images. Ces techniques de masquage des erreurs (*error concealment*) sont couramment utilisées dans les transmissions modernes et une description plus précise est donnée au chapitre 3. Le mode ASO possède cependant un inconvénient majeur : il réduit significativement le gain de codage de la vidéo car les corrélations spatiales et temporelles entre les *slices* ne sont pas exploitées durant la phase de compression.

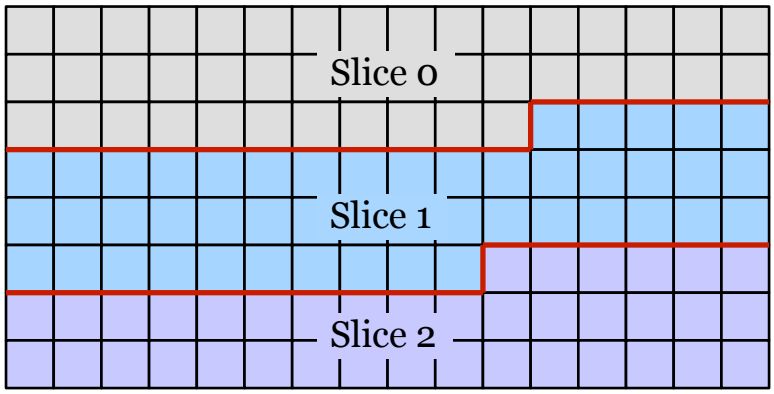


FIG. 1.19 – Découpage d'une image en trois *slices*

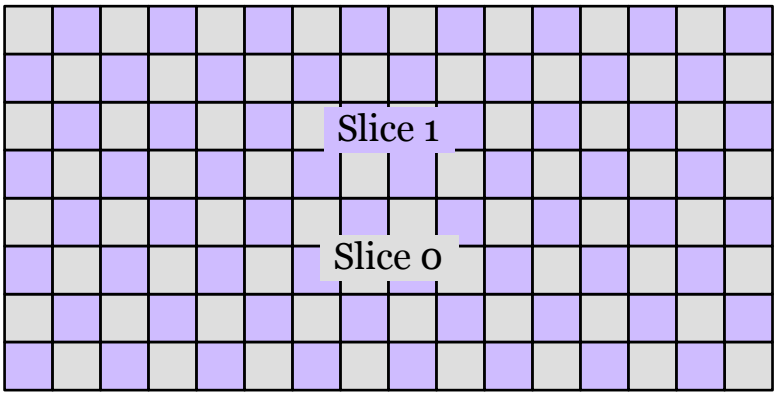


FIG. 1.20 – Découpage d'une image en deux *slices* entrelacés

L'efficacité de la méthode de masquage des erreurs peut être renforcée lorsque le mode ASO est combiné au FMO (*Flexible Macroblock Ordering*) [14]. Dans ce cas, un *slice* n'est pas composé d'un ensemble ordonné de macroblocs et les différents *slices* de l'image peuvent être entrelacés. En résumé, un macrobloc n'est pas associé à un *slice* spécifique en fonction de sa position dans l'image. Une illustration de ce procédé est illustrée sur la figure 1.20. Dans cet exemple, les macroblocs de l'image sont répartis en deux *slices* entrelacés. Il est évident que ce schéma d'entrelacement favorise la reconstruction des zones perdues des images. En contrepartie, il détériore considérablement l'efficacité de l'opération d'encodage.

Le partitionnement des données

Les différents paramètres d'encodage (vecteurs de mouvement, résidus de prédiction, données de contrôle), qui sont transmis dans le flux comprimé, n'ont pas tous le même

impact sur la vidéo décodée. Des recherches pratiques ont prouvé que 80 % de l'énergie des images était restituée par la technique de compensation en mouvement (grâce aux vecteurs de mouvement). Les résidus de prédiction représentent donc une faible proportion de l'énergie du signal (ils permettent d'affiner la qualité vidéo). La norme H.264/AVC tire partie de cette constatation en proposant un mécanisme de séparation des flux (*Data Partitioning* ou DP) [14]. Dans cette technique, les paramètres d'encodage sont répartis en trois catégories qui sont définies ci-dessous :

1. La première classe (DPA) regroupe tous les paramètres d'encodage, excepté les résidus de prédiction. Elle contient principalement les champs contenus dans l'en-tête du flux encodé (*Image header*) ainsi que les paramètres associés à l'étage de prédiction (mode Intra, vecteurs de mouvement, image de référence) de chaque macrobloc.
2. La deuxième catégorie (DPB) rassemble les résidus de prédiction des macroblocs codés en mode Intra.
3. La troisième catégorie (DPC) contient les résidus de prédiction des macroblocs codés en Inter.

Les données encodées d'une image (ou d'un *slice*) sont réparties dans ces trois partitions. Chaque partition est encapsulée dans un paquet NAL particulier. Des niveaux de protection différents sont associés aux données contenues dans chaque partition. La partition DPA est généralement transmise sur un canal fiable pour être reçue par tous les utilisateurs. En revanche, les partitions DPB et DPC sont moins protégées et ne sont pas obligatoirement traitées par tous les récepteurs. Cette méthode permet donc de partager efficacement les ressources du réseau de manière à garantir un service continu à tous les utilisateurs.

1.4 Performances du H.264/AVC

Avant d'exposer les performances d'encodage, il est nécessaire de définir les quatre profils fournis par la norme H.264/AVC. Un profil définit l'ensemble des outils que peut utiliser un encodeur pour comprimer une vidéo. Un décodeur souhaitant décoder cette vidéo devra intégrer un profil équivalent. Le choix du profil dépend du type de l'application. Les quatre profils possibles sont énoncés ci-dessous :

- Le profil de base (*Baseline profile*) propose des outils très simples, qui nécessitent peu de ressources. Il est destiné aux transmissions radio-mobiles et aux dispositifs de visioconférence.
- Le profil principal (*Main profile*) est prévu pour les applications grand public de diffusion et de stockage. Il intègre des outils plus complexes nécessitant des ressources de calcul conséquentes.
- Le profil étendu (*Extended profile*) est adapté à la diffusion des flux vidéo sur les canaux bruités. Il incorpore des outils robustes et des traitements permettant de compenser les variations de débit.

– Le profil haute définition (*High profile*) a été finalisé en 2005. Il est destiné aux applications professionnelles où les images sont très larges et la qualité doit être excellente. Il exploite toutes les options complexes de la norme et nécessite des ressources significatives.

Nous allons maintenant comparer la norme H.264/AVC avec les anciens standards. Dans ce qui suit, l'efficacité du codage est évaluée en déterminant la moyenne du débit binaire pour un PSNR constant (1.2). Pour réaliser les simulations, des séquences vidéo au format CIF ont été utilisées.

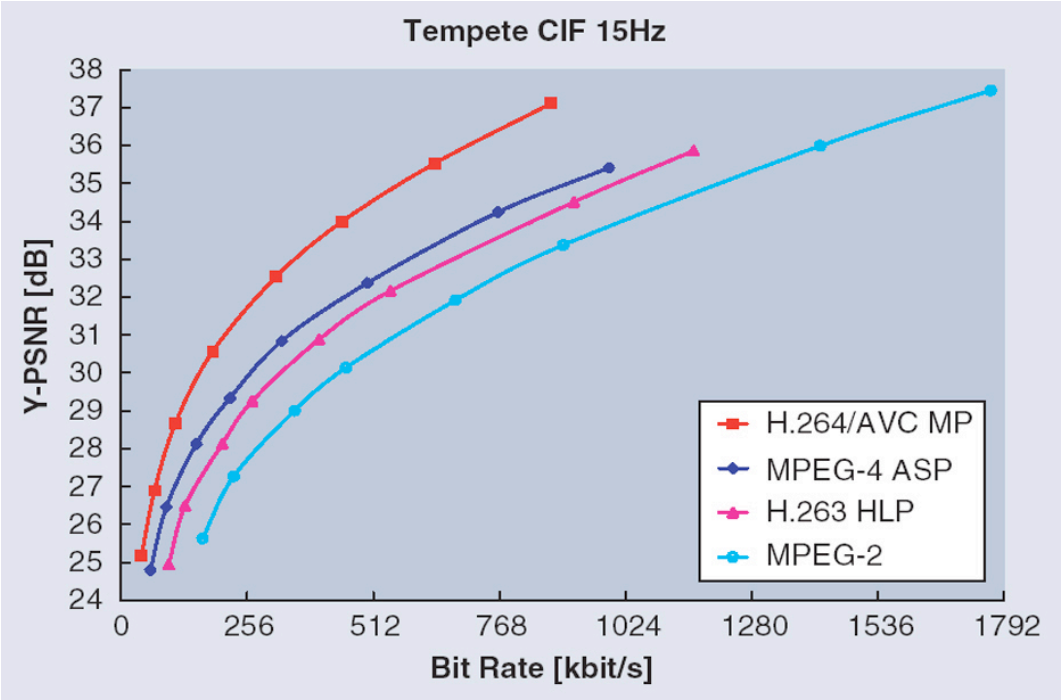


FIG. 1.21 – Evolution du PSNR en fonction du débit pour le profil principal

Pour les applications temps-réel, quatre techniques sont considérées : le H.264/AVC MP (*Main profile*), le MPEG-4 Visual ASP, le H.263 HLP et le MPEG-2 Video ML@MP. La figure 1.21 illustre l'évolution du PSNR de la luminance en fonction du débit binaire pour la séquence *Tempete* encodée à 15 Hz. Nous pouvons constater que le H.264/AVC surpasse les autres encodeurs. Pour une même qualité, il permet d'obtenir des réductions de débit de 63 % par rapport au MPEG-2 et d'environ 37 % par rapport au MPEG-4.

Concernant les dispositifs de visioconférence, le H.264/AVC BP (*Baseline Profile*), le MPEG-4 Visual SP, le H.263 Baseline et le H.263 CHC sont analysés dans cette étude. La figure 1.22 illustre l'évolution du PSNR en fonction du débit pour la séquence *Paris* encodée à 15 Hz. Nous pouvons noter que la nouvelle norme dépasse encore les autres standards en affichant des réductions de débit de l'ordre de 40 % par rapport au H.263 Baseline et de 27 %

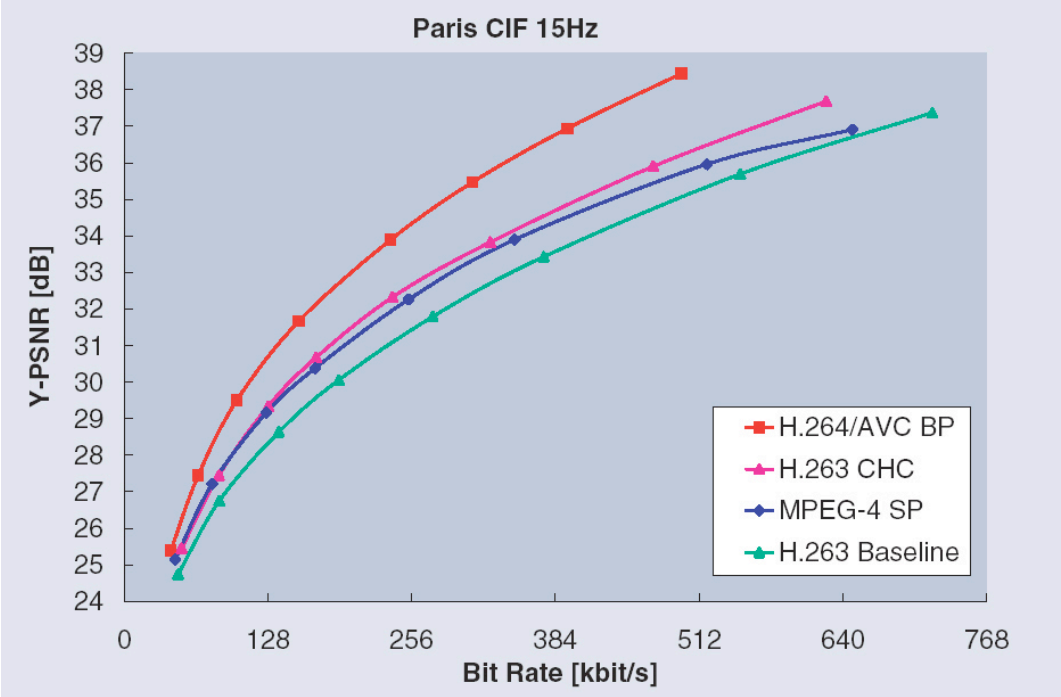


FIG. 1.22 – Evolution du PSNR en fonction du débit pour le profil de base

par rapport au H.263 CHC.

Chapitre 2

Les réseaux mobiles

Dans ce chapitre, nous décrivons le principe de fonctionnement des réseaux mobiles, et plus particulièrement, des protocoles intervenant au niveau d'un récepteur WiFi. Une description précise de la structure des paquets protocolaires est également fournie. Ces informations seront nécessaires dans les chapitres 3 et 4 pour extraire les redondances inhérentes du flux reçu qui seront ensuite utilisées par les traitements robustes.

2.1 Historique

Un réseau représente un ensemble d'équipements reliés entre eux et pouvant se transmettre des informations. Une présentation détaillée de l'évolution des réseaux peut être trouvée dans [22].

Les réseaux peuvent se classer en trois grandes catégories selon le secteur industriel concerné : les réseaux informatiques, de communication et de diffusion. Chacun de ces réseaux possède une histoire différente, propre à ses besoins. Les réseaux informatiques ont été conçus initialement pour transporter des fichiers entre deux ordinateurs, les réseaux de communication pour acheminer des voies téléphoniques et les réseaux de diffusion pour distribuer des canaux multimédias (tels que la télévision ou la radio). Ces différents types de transmission peuvent être rattachés à trois classes de service distinctes :

- Les services de transfert (*download*) qui consistent à transmettre des données entre deux ordinateurs distants. Dans ce genre de transmission, les contraintes de délai sont généralement faibles mais l'intégrité des données reçues est essentielle.
- Les services conversationnels (*real-time*) qui permettent à des utilisateurs distants de dialoguer entre-eux. Ce type de communication est soumise à des contraintes de délai forte et la qualité du signal perçu doit être maximale.
- Les services de diffusion (*broadcast*) dans lesquels un émetteur transmet une même information à un ensemble de récepteurs. Dans cette situation, la communication s'ef-

fectue dans un seul sens, les contraintes de délai sont généralement importantes et la qualité des informations reçues doit être suffisante.

Chacun de ces services possède donc des caractéristiques particulières et nécessite un schéma de transmission spécifique. Cette constatation explique l'évolution indépendante des trois catégories de réseau.

Bien que reposant initialement sur des technologies différentes, ces trois types de réseau ont cependant commencé à converger vers un réseau unique, cohérent et efficace. Le principal facteur, après le développement d'Internet en 1974, fut la généralisation du numérique dans tous les dispositifs électroniques. L'aboutissement de la norme GSM [23] en 1989 fit progresser le secteur des télécommunications d'un grand pas sur plusieurs axes technologiques : les communications téléphoniques deviennent numériques et intègrent la mobilité (transport hertzien de l'information). L'évolution du GSM, à travers les normes GPRS [24], UMTS [25] puis WIMAX [26], accélère progressivement la convergence entre le monde Internet et les télécommunications mobiles.

Parallèlement, les réseaux de diffusion ont connu une grande avancée suite à l'élaboration de la norme DVB [27] en 1993. Avec DVB, le numérique détrône la télévision analogique en réduisant les débits requis de chaque chaîne (compression vidéo) et en fournissant des schémas de transmission paquétisée pour des supports physiques divers (terrestre, satellite, câble, portable).

En résumé, l'ère du numérique a permis d'uniformiser les réseaux de transmissions et a favorisé la mobilité des usagers. Dans un futur proche, Internet centralisera toutes les communications numériques (transmission paquétisée de type IP) et les abonnés accéderont librement au réseau grâce aux avantages du sans-fil.

2.2 Schéma général

Internet est un réseau informatique qui permet d'interconnecter un ensemble de terminaux à l'échelle mondiale. Le transfert des données entre deux terminaux s'effectue sous forme de paquets. Chaque terminal possède une adresse unique (appelée adresse IP) qui dépend de sa localisation (opérateur, réseau local, etc). Chaque paquet transmis sur le réseau contient l'adresse IP du terminal source ainsi que l'adresse IP du terminal destination. L'acheminement des paquets entre l'émetteur et le récepteur s'effectue par le biais de routeurs. Les routeurs sont les noeuds du réseau et centralisent un ensemble de chemins.

Quand un routeur reçoit un paquet, il analyse son adresse IP de destination et transmet le paquet sur le chemin le plus approprié. Après avoir traversé un ensemble de routeurs, le paquet arrive à sa destination. Pour des raisons de fiabilité, Internet est un réseau fortement maillé : pour parvenir à destination, l'information peut emprunter des chemins multiples. Par conséquent, tous les paquets d'un même ensemble d'information n'empruntent pas forcément le même chemin pour parvenir à destination et n'arrivent pas nécessairement dans l'ordre

dans lequel il ont été émis. La présence de l'adresse IP de la source dans chaque paquet permet au récepteur de répondre à l'émetteur.

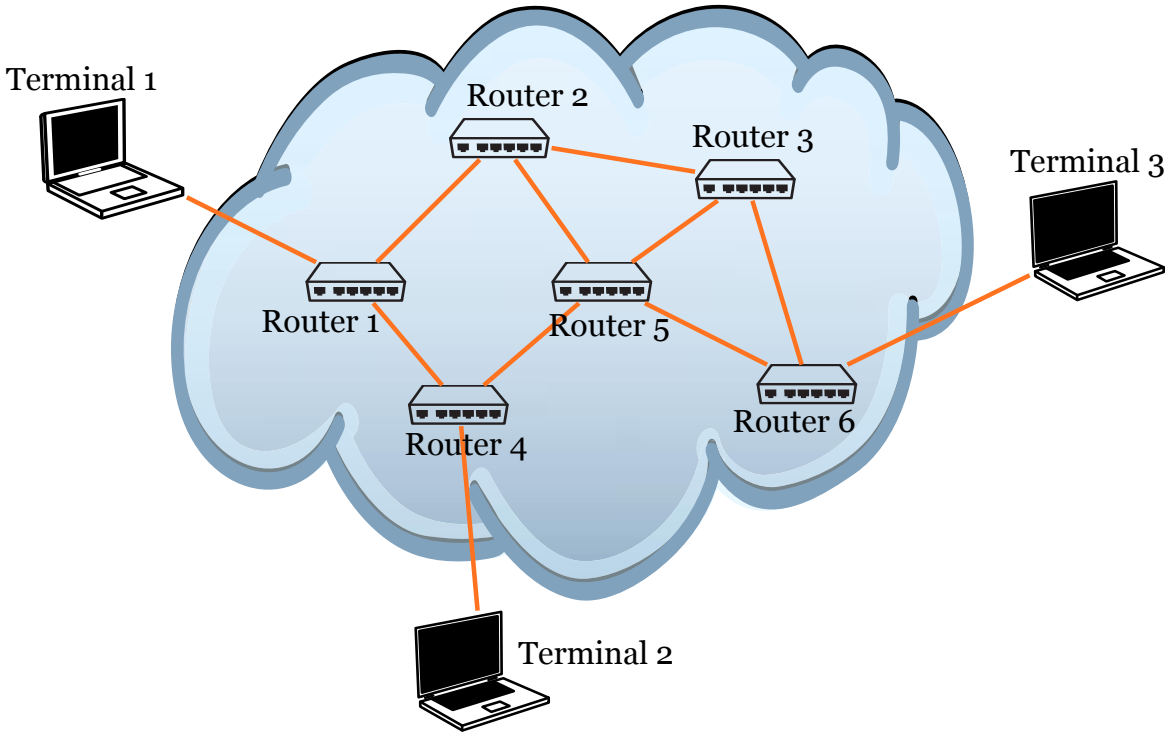


FIG. 2.1 – Schéma simplifié du réseau Internet

La figure 2.1 illustre une représentation simplifiée d'Internet. Nous pouvons constater qu'une multitude de chemins permettent de relier les trois terminaux. Le rôle des routeurs est d'acheminer efficacement les paquets d'un terminal à un autre. Actuellement, deux types de support physique assurent le transport des données entre les routeurs : la fibre optique et le câble.

La figure 2.2 présente les supports physiques prédominants du réseau Internet ainsi que leurs débits respectifs. Les supports câblés permettent d'atteindre des débits de 100 Mbps avec des taux d'erreurs binaires négligeables (de l'ordre de 10^{-6}), tandis que la fibre optique offre de très hautes capacités de transmission (jusqu'à 40 Gbps par fibre) avec des taux d'erreurs mille fois plus faibles (*Bit Error Rate* ou BER inférieur à 10^{-9}). De telles caractéristiques expliquent la tendance des opérateurs à remplacer progressivement le câble par la fibre. Par ailleurs, la congestion liée aux files d'attente à l'entrée des routeurs ne cesse de diminuer avec l'évolution de la puissance des processeurs et l'accroissement de la capacité de stockage des mémoires. Par conséquent, la fiabilité du transport (taux d'erreurs et congestions faibles) combinée aux vitesses de transmission élevées permettent aux utilisateurs finaux de disposer de débits utiles avoisinant les 10 Mbps. Avec de tels débits, même les services les plus contraignants (comme la visioconférence) deviennent envisageables.

Dénomination	Type de support	Débit approximatif
DS0	fils de cuivre torsadés	64 kbps
DS1/T-1	fils de cuivre torsadés	1.5 Mbps
DS3/T-3	fils de cuivre torsadés	45 Mbps
E-1	câble coaxial	2 Mbps
E-3	câble coaxial	35 Mbps
ADSL réception	fil téléphonique	8 Mbps
ADSL émission	fil téléphonique	1 Mbps
ADSL 2+ réception	fil téléphonique	25 Mbps
ADSL 2+ émission	fil téléphonique	1 Mbps
VDSL réception	fil téléphonique	52 Mbps
VDSL émission	fil téléphonique	12 Mbps
VDSL 2 réception	fil téléphonique	100 Mbps
VDSL 2 émission	fil téléphonique	100 Mbps
OC-1	fibre optique	52 Mbps
OC-3	fibre optique	156 Mbps
OC-9	fibre optique	467 Mbps
OC-12	fibre optique	622 Mbps
OC-48	fibre optique	2.5 Gbps
OC-255	fibre optique	13 Gbps
OC-768	fibre optique	40 Gbps

FIG. 2.2 – Débits possibles pour différents types de supports physiques

L'attrait des utilisateurs pour la mobilité a débuté avec le développement des premières normes de transmission radio numériques. Ces supports hertziens ont un intérêt significatif aux extrémités des réseaux car ils permettent aux utilisateurs de rester connectés à l'intérieur d'une zone géographique plus ou moins étendue. Les normes radio-mobiles sont classées en fonction de leur zone de couverture, appelée cellule. Les plus populaires sont représentées sur la figure 2.3. Les réseaux WPAN (*Wireless Personal Area Network*) ont une portée de seulement quelques mètres et permettent d'atteindre des débits allant de 1 à 100 Mbps avec la technologie Bluetooth [28]. Une couverture plus importante (d'une centaine de mètres) est obtenue avec les réseaux WLAN (*Wireless Local Area Network*), notamment avec la norme 802.11*n* [29] où les débits peuvent dépasser 250 Mbps. A des échelles plus importantes (d'une dizaine de kilomètres), la norme 802.16*e* [30, 31] peut fournir des débits de 20 Mbps. Cette norme caractérise les réseaux WMAN (*Wireless Metropolitan Area Network*).

Contrairement aux supports câblés qui acheminent un signal entre un émetteur et un récepteur unique, les liens radio-mobiles diffusent les informations à tous les utilisateurs contenus dans la cellule. En somme, les ondes hertziennes sont difficiles à confiner dans une zone géographique restreinte. Cette propriété est à la base de la mobilité. Elle possède néanmoins un inconvénient majeur : les ressources radio doivent être partagées entre tous les récepteurs présents dans la cellule. Les plages de fréquences radio étant limitées, les débits indicatifs fournis dans le paragraphe précédent sont à diviser par le nombre d'abonnés en communication à un instant donné dans la cellule considérée. Le débit alloué à chaque

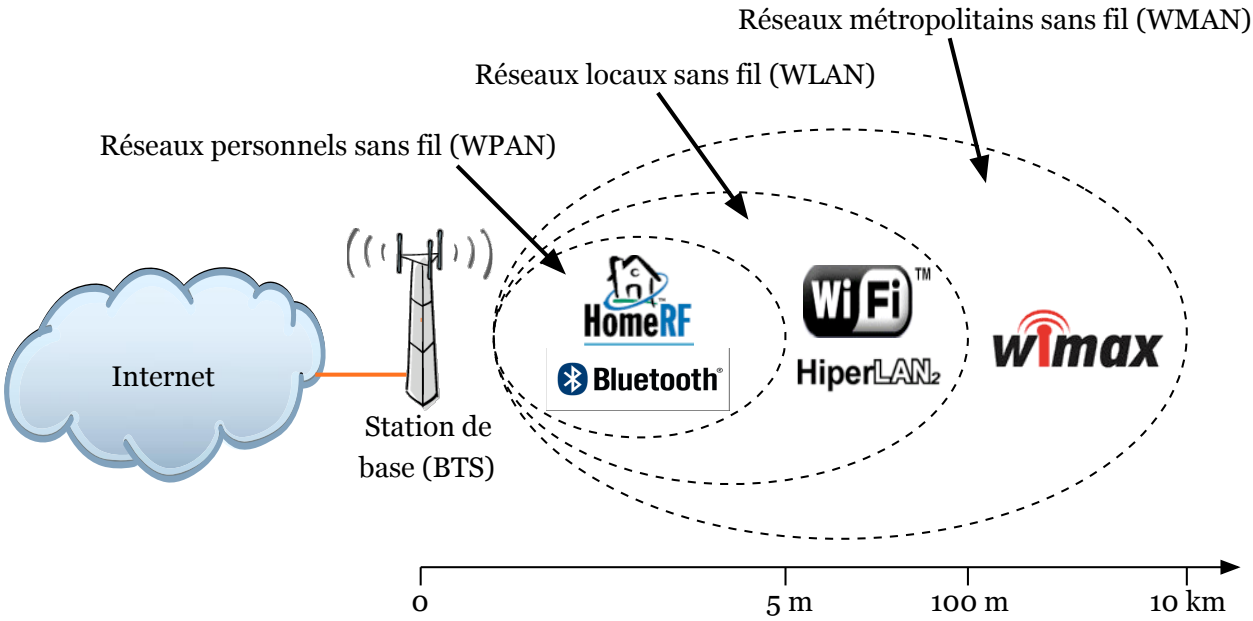


FIG. 2.3 – Normes associées aux différentes catégories de réseaux sans fil

utilisateur est donc d'autant plus faible que la zone de couverture est large (pour une densité de population constante). Dans le cas de WIMAX, par exemple, il est fréquent que le débit n'excède pas 100 kbps par abonné.

Par ailleurs, les canaux radio sont soumis à de fortes perturbations qui détériorent la qualité des informations reçues (BER élevé). Pour restituer les informations, des mécanismes de protection sont intégrés dans les récepteurs mobiles. Une description détaillée de ces mécanismes est fournie dans la partie 2.3. Pour l'instant, retenons juste que deux méthodes sont généralement employées. La première consiste à retransmettre les paquets qui ont été détectés erronés par le récepteur. Dans la deuxième technique, l'émetteur ajoute des redondances aux données à transmettre. Ces informations complémentaires sont ensuite utilisées par le récepteur pour corriger les erreurs de transmission. La combinaison de ces deux techniques permet de compenser efficacement le taux d'erreurs importants survenant lors des transmissions radio. En contrepartie, ces solutions réduisent la bande-passante réservée pour le transfert des informations utiles. Si nous reprenons l'exemple de WIMAX, le débit utile dépasse rarement 50 kbps par utilisateur.

En résumé, le réseau coeur d'Internet possède une structure maillée dont les noeuds sont représentés par des routeurs et les jonctions par des lignes optiques ou électriques. Les informations sont transmises sous forme de paquets. Les paquets contiennent un identifiant unique caractérisant le destinataire : l'adresse IP. Cette adresse permet aux routeurs d'aiguiller les paquets vers leur destination finale. Cette architecture permet d'atteindre des débits de transmission élevés avec une grande fiabilité. Aux extrémités du réseau, la mobilité des usagers est assurée grâce aux ondes hertziennes. Les débits utiles fournis par les canaux radio sont

cependant très faibles et ne permettent pas d'acheminer efficacement des données multimédia telles que la vidéo. En définitif, le support radio représente actuellement le maillon critique de la chaîne de transmission complète.

Le champ d'étude couvert par cette thèse porte exclusivement sur les problématiques associées à la liaison hertzienne. Nous négligeons donc les problèmes liés au fonctionnement du réseau central pour nous concentrer uniquement sur la partie radio. Pour restreindre la diversité du contexte, nous limitons notre étude à la transmission temps-réel de données vidéo entre un serveur relié à Internet et un terminal mobile. Le coeur du réseau (Internet) étant supposé idéal, nous considérons que la station de base reçoit correctement et rapidement toutes les informations transmises par le serveur. L'objectif de cette thèse vise à améliorer la qualité de la transmission entre la station de base et le terminal. Les uniques points d'action se limitent au serveur, à la station de base et au récepteur.

2.3 Architecture des réseaux mobiles

Pour transmettre efficacement les données du serveur vers le terminal, il est nécessaire de définir une architecture logicielle capable de s'adapter aux différents supports de transmission. Cette architecture repose sur un modèle en couches. La référence universelle correspond au modèle OSI (*Open Systems Interconnection*), illustré sur la figure 2.4. Ce modèle est constitué d'un empilement de sept couches qui définissent les règles communes de communication et de coopération entre les équipements. Chaque couche dialogue avec son homologue par l'intermédiaire d'un protocole. De manière simpliste, un protocole permet de réaliser une liaison virtuelle entre les couches complémentaires. Par ailleurs, chaque couche fournit des informations à la couche supérieure ou inférieure grâce à des primitives de service.

L'acheminement des données entre l'émetteur et le récepteur est assuré par les couches basses 1, 2, 3 et 4. Chacune de ces couches est rattachée à un équipement particulier et agit à un niveau spécifique. En revanche, les couches hautes 5, 6 et 7 réalisent le traitement des informations et font uniquement intervenir le serveur et le terminal. Les couches du modèle OSI possèdent donc des fonctions clairement définies. Le rôle de chaque couche est décrit brièvement ci-dessous :

1. La couche Physique assure le transport physique des données. Elle transforme le flux binaire à transmettre en signal analogique, adapté au canal de transmission (câble, fibre optique, onde radio). Les spécifications incluent généralement une opération de codage canal destinée à protéger les informations des erreurs de transmission. Cette couche assure également la synchronisation des données entre l'émetteur et le récepteur.
2. La couche Liaison gère les communications entre deux équipements adjacents, directement reliés entre eux par le support physique. Elle répartit notamment les informations à transmettre en fragments, dont la taille est adaptée au canal. Elle contrôle l'intégrité des fragments reçus en incorporant des mécanismes de détection d'erreurs. Elle commande la retransmission des fragments erronés. Elle travaille conjointement avec la

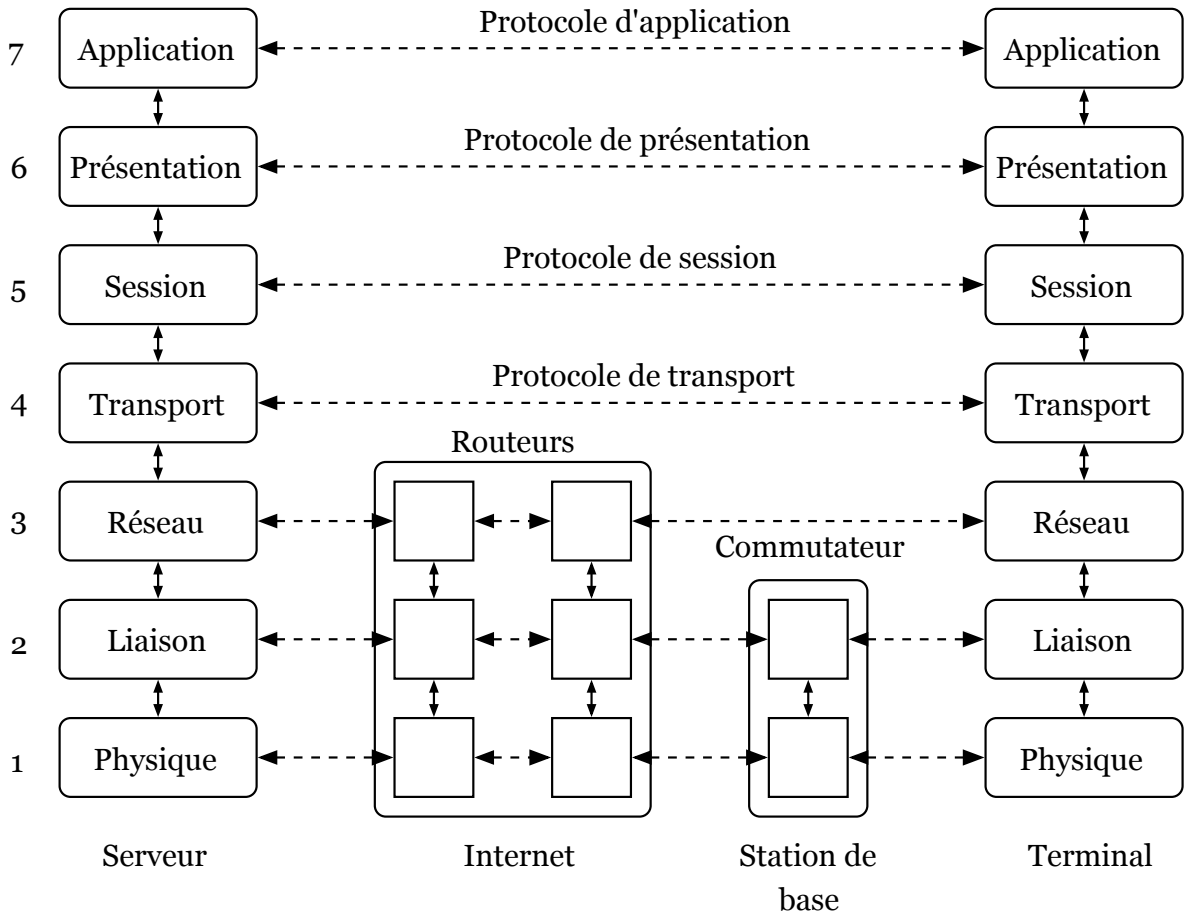


FIG. 2.4 – Schéma de transmission s'appuyant sur le modèle OSI

couche physique pour réaliser le multiplexage des utilisateurs lorsque le support est partagé, elle se charge de l'allocation des ressources de transmission.

3. La couche Réseau établit une voie de communication entre le serveur et le terminal, elle participe au routage des informations. Cette opération est exécutée par les routeurs. Cette couche est la seule à être directement concernée par la topologie du réseau. Elle est également la dernière couche supportée par les équipements du réseau central, les couches supérieures sont réalisées uniquement dans les dispositifs d'extrémité.
4. La couche Transport se charge de l'acheminement des données de bout en bout entre les processus. Elle supervise la transmission à un niveau global. Elle effectue la transition entre la transmission des données sur le réseau et le service que désire recevoir un utilisateur. Elle réalise le multiplexage des données entre les processus. Elle fragmente les données issues des couches supérieures en paquets, dont la taille est adaptée au réseau. Deux types de service sont disponibles : les services en mode connecté ou non-connecté. La sélection du service dépend de la nature des données à transmettre (transfert de fichier ou service conversationnel). En mode connecté, la couche Transport garantit la qualité du transport : pas de perte de paquets et intégrité des données reçues. Ce mode

peut néanmoins engendrer des délais de propagation élevés. En mode non-connecté, la transmission n'est pas assurée et certains paquets peuvent être perdus (congestion du réseau, corruption des données). Les paquets sont transmis une seule fois par l'émetteur et le récepteur final n'acquitte pas les paquets. La durée de transmission est donc faible mais la réception non garantie.

5. La couche Session organise et synchronise les échanges entre les tâches distantes. Elle établit une liaison entre deux programmes devant coopérer et commande leur dialogue (gestion du jeton). Elle permet aussi de gérer les communications multipoints, telles que la diffusion d'un message. Suite à un incident technique (panne sur le réseau), la couche Session permet de rétablir le dialogue en utilisant des marqueurs de reprise insérés dans le flux de données.
6. La couche Présentation garantit la compatibilité des informations entre les tâches communicantes. Elle est chargée du codage des données applicatives. Elle peut typiquement comprimer, crypter ou convertir les données. Elle réalise la transition entre le format des données utilisateur et le flux transporté par le réseau.
7. La couche Application représente le point de contact entre l'utilisateur et le réseau. Elle fournit donc aux utilisateurs les services de base offerts par le réseau.

Bien que le modèle OSI soit la structure réseau la plus étudiée, elle n'a pas su s'imposer sur le marché. La raison de son échec est principalement liée à sa complexité. Lors du développement d'Internet, les industriels ont préféré utiliser le modèle TCP/IP. Cette architecture a repris l'approche modulaire du modèle OSI en se limitant à quatre couches protocolaires. L'acronyme TCP/IP désigne le nom des deux premiers protocoles de cette architecture : le protocole IP [32] de la couche Réseau et le protocole TCP [33] de la couche Transport. Sa popularité résulte de son histoire : contrairement au modèle OSI, l'architecture TCP/IP est née d'une implémentation. La procédure de normalisation est venue ensuite. Son principal atout est donc associé à sa simplicité. La comparaison entre les deux modèles est illustrée sur la figure 2.5.

Contrairement au modèle OSI, l'architecture TCP/IP ne possède pas de couches Session et Présentation. Les fonctions assurées par ces deux couches sont directement intégrées dans la couche Application. Par ailleurs, les couches Physique et Liaison sont rassemblées dans la couche Hôte-réseau. L'implémentation de cette couche n'est pas spécifiée dans le modèle TCP/IP. Seule son rôle est défini : elle doit permettre d'accéder au support physique et d'envoyer des paquets IP sur le réseau. De manière concrète, la norme requise dépend de la technologie employée sur le réseau local (Ethernet, WiFi).

Comme nous l'avons précisé auparavant, notre étude se limite à la transmission temps-réel (*streaming*) de données vidéo entre un serveur et un terminal mobile. Dans ce type de transmission, le transfert d'informations entre l'émetteur et le récepteur doit être très rapide, empêchant tout dialogue entre les interlocuteurs situés aux extrémités. Le protocole TCP, fournissant un service orienté connexion, n'est donc pas adapté à notre situation. Il est classiquement remplacé par deux protocoles complémentaires qui forment la couche UDP/RTP [34, 35]. Par ailleurs, nous focalisons sur les problématiques associées au support de transmission hertzien situé entre la station de base et le terminal. A ce niveau, la couche

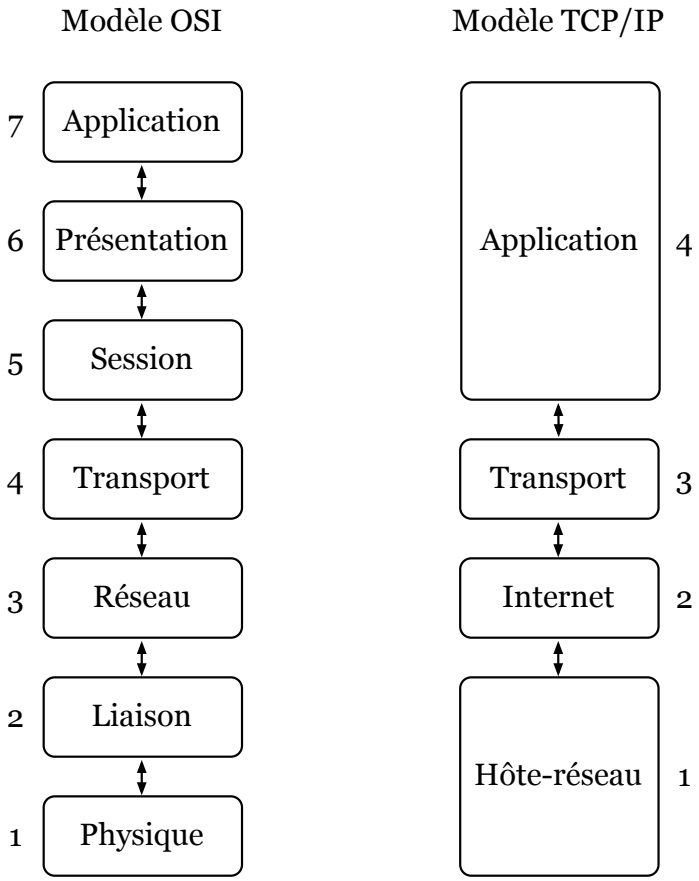


FIG. 2.5 – Comparaison entre l'architecture TCP/IP et le modèle OSI

Hôte-réseau doit donc correspondre à une norme de propagation radio-mobile. Dans cette étude, nous considérons que le réseau d'extrémité exploite la technologie sans fil 802.11 [36]. Le choix de cette norme s'explique pour plusieurs raisons : sa simplicité d'implémentation, sa popularité (matériel existant, large documentation) et sa proximité (compatibilité) avec les protocoles Internet. La norme 802.11 fournit les spécifications relatives à l'accès au support radio, c'est-à-dire l'implémentation de la couche Physique (PHY) et de la couche Liaison (MAC). La chaîne de transmission complète est représentée sur la figure 2.6.

Dans les paragraphes qui suivent, nous définissons les différents protocoles intervenant directement entre le terminal et les divers équipements du réseau (point d'accès WiFi, dernier routeur et serveur). Etant donné que les travaux introduits dans cette thèse sont destinés à améliorer la qualité de la vidéo décodée au niveau du récepteur, le reste de la chaîne de transmission sera transparent et considéré idéal. Les protocoles sont représentés sur la figure 2.7. Au niveau du récepteur, la pile protocolaire correspond donc à l'empilement des couches : PHY 802.11/MAC 802.11/IP/UDP/RTP. Des mécanismes d'encapsulation et de fragmentation des données sont répartis à travers toutes les couches de la pile.

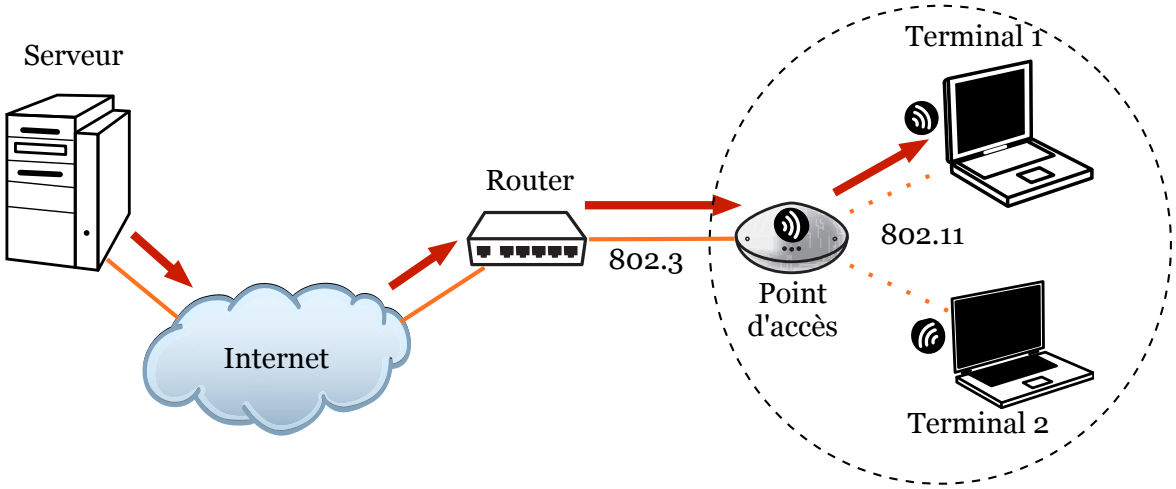


FIG. 2.6 – Schéma de transmission général

2.3.1 La couche Physique 802.11 (PHY)

Les mécanismes de propagation sont des éléments déterminants dans la transmission radio. L'environnement peut fortement modifier les caractéristiques du signal transmis. En propagation directe, le signal capté par l'antenne du récepteur est juste atténué d'un facteur dépendant de la distance entre l'émetteur et le terminal. En milieu confiné, les conditions de propagation sont nettement moins déterministes. Les ondes sont réfléchies par des obstacles environnants et le signal reçu par l'antenne du récepteur correspond à une somme d'échos atténués et retardés. Les trajets multiples des ondes créent des interférences locales et détériorent la qualité du signal reçu. Par ailleurs, la mobilité des usagers et le changement de la topologie physique (tel que l'ouverture d'une porte) modifient les caractéristiques du canal de transmission (la fonction de transfert du canal varie dans le temps). Le bruit électromagnétique (généré par un four à micro-ondes) peut aussi accentuer la déformation du signal reçu. Ces phénomènes sont illustrés sur la figure 2.8. Toutes ces perturbations détériorent la qualité de la transmission et génèrent des erreurs binaires lors de la réception.

Dans cette partie, nous nous focalisons sur les mécanismes intervenant dans la couche Physique de la norme 802.11. Ces outils doivent permettre de transmettre les données entre le point d'accès et le terminal en s'adaptant aux contraintes sévères du canal hertzien. Ces mécanismes visent à minimiser la consommation des ressources radio tout en garantissant la qualité de réception. La couche Physique se charge aussi d'évaluer l'état de disponibilité du canal (libre ou occupé). Elle transmet un CCA (*Clear Channel Assessment*) à la couche Liaison lorsque le canal est libre.

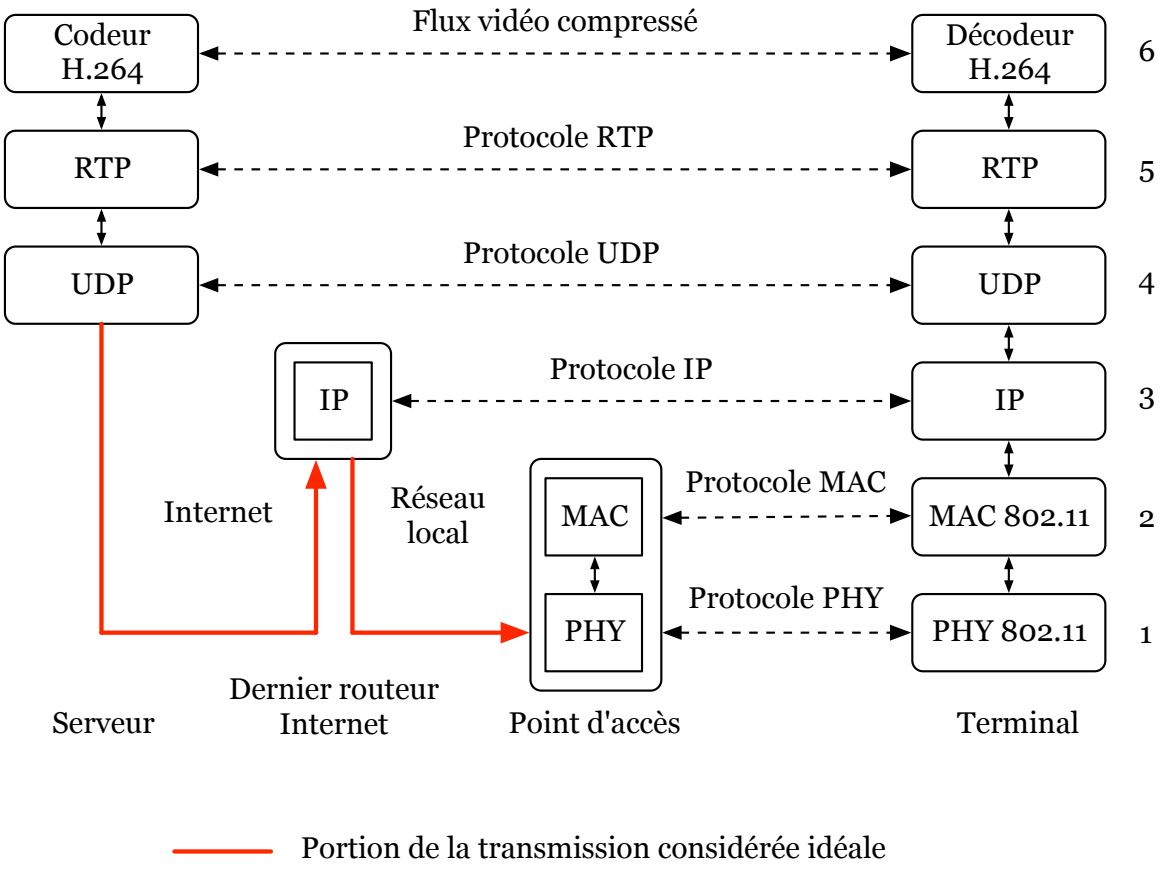


FIG. 2.7 – Illustration des protocoles intervenant entre le terminal et les autres équipements

Description technique

La couche Physique de la norme 802.11 fournit des débits de 1 ou 2 Mbps dans la bande spectrale des 2.4 GHz. Deux techniques d'étalement de spectre sont définies dans le standard : le FHSS (*Frequency Hopping Spread Spectrum*) et le DSSS (*Direct Sequence Spread Spectrum*).

Le FHSS Cet outil correspond à une méthode d'étalement de spectre par saut de fréquence. Plus précisément, l'émetteur utilise successivement plusieurs fréquences de porteuse, selon une séquence connue uniquement de l'émetteur et du récepteur. Ces changements synchronisés sont définis par un code, qui spécifie l'amplitude des sauts. Pour permettre ce fonctionnement, la bande utilisée doit auparavant être découpée en spectres étroits. Les spécifications de la norme 802.11 proposent de diviser la bande des 2.4 GHz en 79 canaux de 1 MHz. La transmission d'informations sur chaque canal peut durer de quelques dizaines à quelques centaines de millisecondes (généralement 300 à 400 ms). Cette méthode permet d'atteindre des débits de base de 1 Mbps. Par ailleurs, le signal généré n'est pas sensible aux interférences

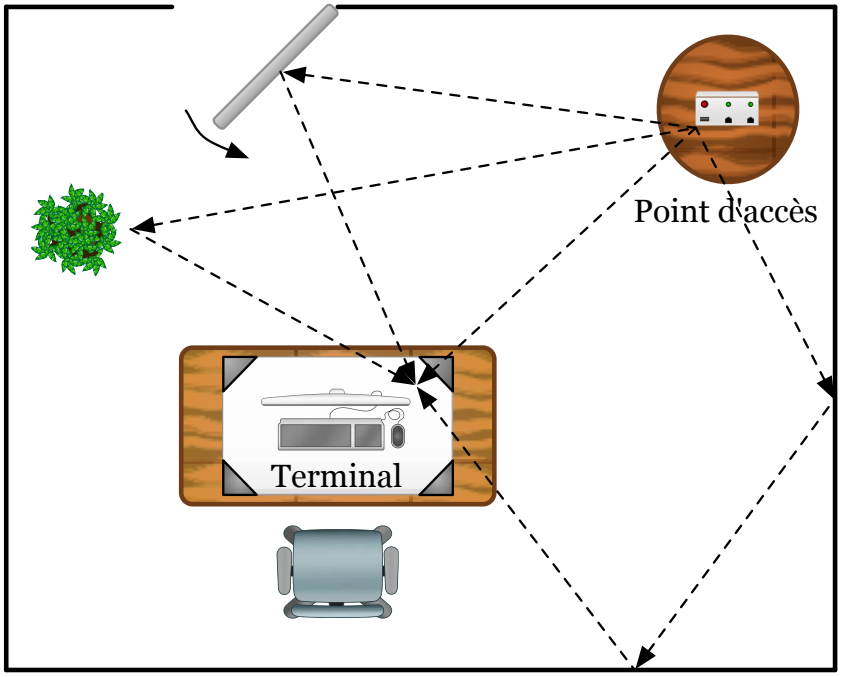


FIG. 2.8 – Transmission des ondes radio dans un environnement confiné

à bande étroite (grâce à la diversité en fréquence). La figure 2.9 illustre le fonctionnement du FHSS.

Le FHSS est combinée à une modulation GFSK (*Gaussian Frequency Shift Keying*), dans laquelle les niveaux significatifs sont représentés par des fréquences décalées par rapport à la porteuse de base. A 1 Mbps, les bits sont transmis successivement et sont modulés séparément par un modulateur 2-GFSK. Les deux états binaires (0 et 1) sont définis par deux fréquences distinctes, centrées autour de la porteuse et séparées de 320 kHz. A 2 Mbps, deux bits sont envoyés simultanément par le biais d'un modulateur 4-GFSK. Les quatre états binaires sont représentés par quatre fréquences centrées autour de la porteuse et décalées de 144 kHz. Les modulations 2-GFSK et 4-GFSK sont illustrées sur la figure 2.10.

Le DSSS Cette technique représente une méthode d'étalement de spectre, dans laquelle un bit est représenté par une séquence binaire à transitions d'état très rapides. A l'opposé de la technique précédente, le DSSS exploite une bande spectrale continue et étalée. Plus le code est long, plus la fréquence de transition est élevée et plus le spectre est large. Dans la norme 802.11, un code de Barker de 11 bits (*chips*) est utilisé pour étaler le spectre du flux binaire de 1 Mbps. L'état logique bas (0) est représenté par la séquence 10110111000, et l'état haut (1) par son inverse 01001000111. Les chips sont cadencés à une fréquence de 11 MHz et l'énergie du signal en bande de base est donc concentrée dans un lobe primaire de 22 MHz. Cette méthode possède une forte immunité au bruit (par ajout de redondance)

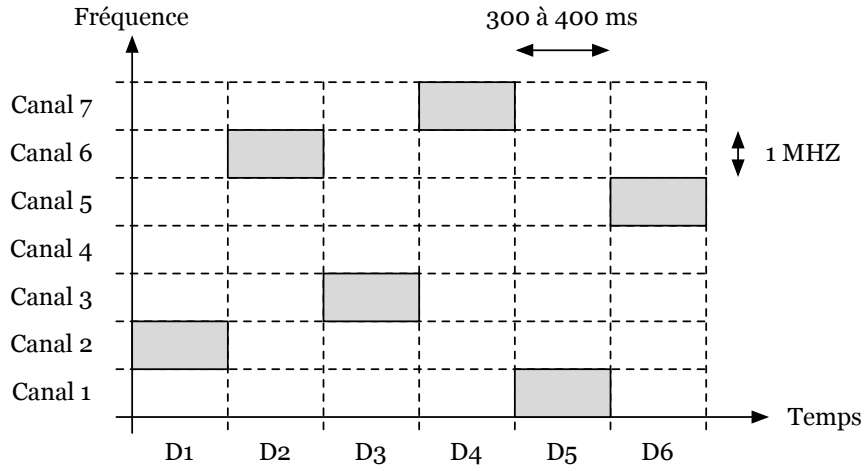


FIG. 2.9 – Principe de la méthode FHSS

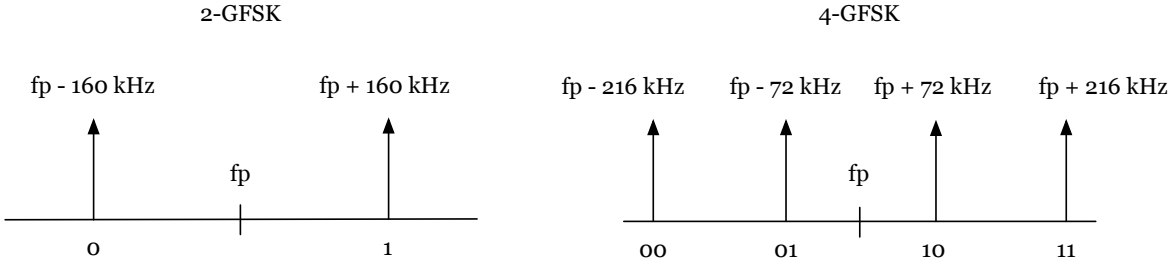


FIG. 2.10 – Illustration des modulations 2-GFSK et 4-GFSK

et permet de renforcer la qualité de la transmission. Le principe d'encodage est illustré sur la figure 2.11.

Avec le DSSS, les informations binaires sont modulées dans la phase du signal de référence (porteuse). Il existe différentes formes de modulation de phase et la norme 802.11 utilise le PSK (*Phase Shift Keying*). Dans ce schéma, la valeur d'un symbole est codée dans la phase. Le mode BPSK fournit des débits de 1 Mbps en encodant les bits successivement. Dans ces conditions, les variations de la phase sont limitées à deux valeurs : 0 et π. Le mode QPSK est employé pour doubler le débit. Il permet de transmettre deux bits simultanément en intégrant quatre valeurs de déphasage : 0, $\frac{\pi}{2}$, π et $\frac{3\pi}{2}$. Les diagrammes de phase associés aux modulations BPSK (*Binary Phase Shift Keying*) et QPSK (*Quadrature Phase Shift Keying*) sont représentés sur la figure 2.12.

Dans ce travail de thèse, nous avons opté pour la méthode de transmission basée sur le DSSS. Deux facteurs nous ont amenés à faire ce choix. Le premier est associé à sa facilité d'implémentation. En effet, le DSSS est avantageusement basé sur des traitements numériques, facilement reproductibles par un ordinateur. A l'inverse, le FHSS repose princi-

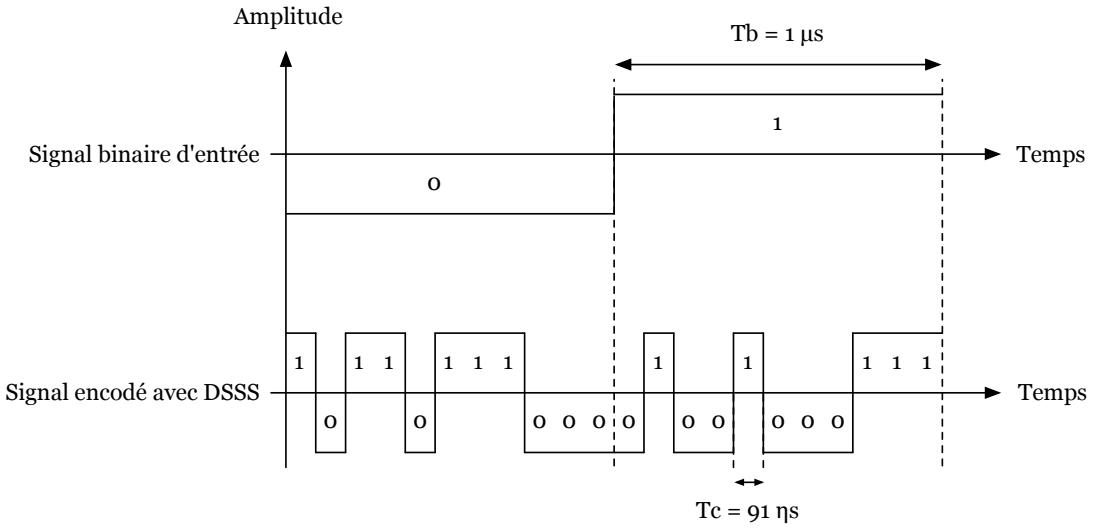


FIG. 2.11 – Méthode d'encodage DSSS avec un code de Barker de 11 bits

palement sur des opérations analogiques (modulation de fréquence) qui sont plus difficiles à simuler. Le deuxième critère de sélection est lié aux performances du DSSS. Contrairement au FHSS, le DSSS intègre des outils de correction d'erreurs (code de Barker) qui renforcent la qualité des signaux reçus et justifient son utilisation.

Remarque 1 *Dans certains cas, il est cependant plus réaliste d'utiliser un codage convolutif (à la place d'un code à étalement). En effet, la majorité des systèmes de correction reposent actuellement sur des codeurs convolutifs. C'est pourquoi, dans certaines simulations, nous avons remplacé le code à étalement de spectre de la norme* 802.11 *par le codeur convolutif utilisé dans la norme* 802.11*a. Le schéma bloc de l'encodeur est illustré sur la figure 2.13. Il consiste à encoder chaque bit entrant en* 2 *bits codés (taux de codage* $R = 1/2$*). Il est composé de* 6 *registres à décalage cadencés par une horloge. Les bits d'information à coder sont présentés séquentiellement à l'entrée du premier registre. A chaque impulsion d'horloge, un décalage global vers la droite est effectué et les* 2 *bits de sortie sont déterminés en fonction des valeurs stockées dans les registres (*ou exclusif*). Chaque bit de sortie dépend d'un polynôme générateur qui sélectionne les bits d'intérêt dans les registres. Concernant ce codeur, les deux polynômes générateurs sont* $g_0 = 133_8$ *et* $g_1 = 171_8$*.*

Format des paquets PHY

Le protocole DSSS permet de transmettre les données binaires venant des couches supérieures sur le canal hertzien situé entre le point d'accès et le terminal. Ces données utiles sont réparties dans des paquets, à l'intérieur d'un champ de contenu (*payload*). Un en-tête (*header*) est ajouté à chaque paquet et contient les paramètres de transmission associés au

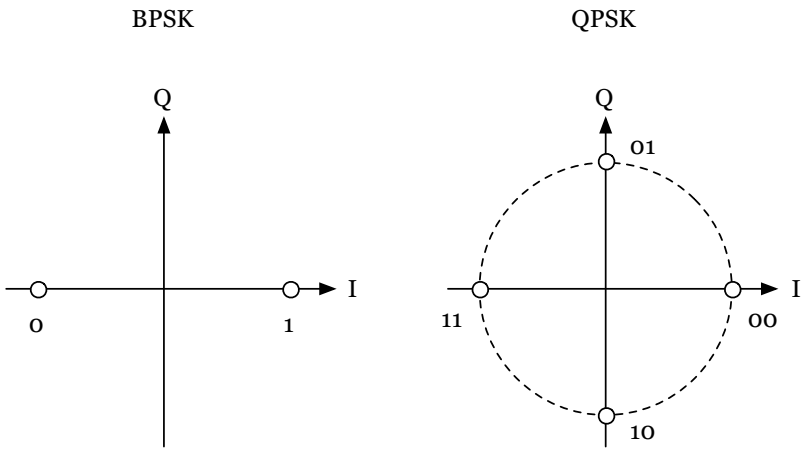


FIG. 2.12 – Diagrammes de phase des modulations BPSK et QPSK

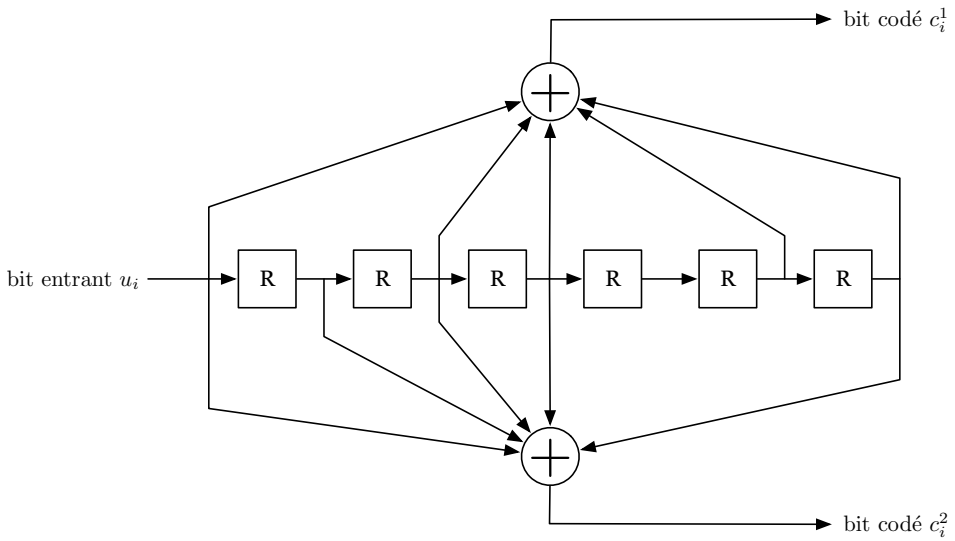


FIG. 2.13 – Codeur convolutif de la norme 802.11*a*

contenu transporté. Par ailleurs, un préambule (*preamble*) est ajouté au début de chaque paquet et permet de synchroniser l'émetteur et le récepteur. Le format des paquets issus de la couche Physique est illustré sur la figure 2.14.

Dans le protocole DSSS, le préambule et l'en-tête des paquets PHY sont toujours transmis à 1 Mbps en utilisant le schéma de modulation le plus robuste, le BPSK. Cette procédure pré-définie permet au récepteur de se synchroniser correctement sur le flux et de récupérer les paramètres associés à la transmission du reste du paquet (*payload*). En revanche, les données utiles sont transférées à un débit variable de 1 ou 2 Mbps. Les champs d'information contenus dans les paquets PHY sont brièvement décrits ci-dessous :

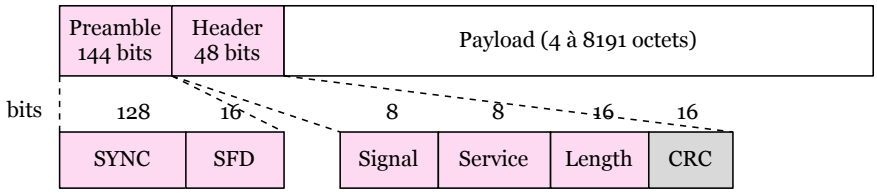


FIG. 2.14 – Format des paquets de la couche Physique 802.11

- Le champ *SYNC* est une séquence constante, composée de 128 bits à 1. Il permet au récepteur de détecter le début d'un message (détection d'énergie sur le canal), de se synchroniser sur la fréquence de la porteuse et d'ajuster le gain de l'étage d'amplification d'entrée.
- Le champ *SFD* est un code constant de 2 octets fixé à $F3A0_{16}$. Le récepteur l'utilise pour se synchroniser temporellement sur le flux binaire.
- Le champ *Signal* de 1 octet spécifie le débit de transmission des données utiles (*payload*). Cette valeur (en base 16) est égale à $0A_{16}$ pour la modulation BPSK et à 14_{16} pour la modulation QPSK.
- Le champ *Service* de 1 octet est réservée pour des recommandations futures. Sa valeur est fixée à 00_{16} dans la norme 802.11.
- Le champ *Length* indique le délai de transmission en microsecondes nécessaire pour envoyer les données utiles. Sa valeur, codée sur 2 octets, est comprise entre 16 et $2^{16}-1$. Elle dépend de deux paramètres : la taille et le débit de la *payload*.
- Un CRC de 2 octets protège les champs *Signal*, *Service* et *Length*. Il correspond à un CCITT CRC-16 et son polynôme générateur est défini par : $G(x) = x^{16} + x^{12} + x^5 + 1$. Il permet au récepteur de détecter des erreurs dans l'en-tête des paquets. Quand une erreur est détectée, le paquet complet est effacé.
- La *payload* contient les données utiles à transmettre : elle représente les informations d'un paquet MAC. Ce champ peut renfermer entre 4 et 8191 octets. Sa taille est facilement déterminée à partir des champs *Signal* et *Length*.

2.3.2 La couche Liaison 802.11 (MAC)

Dans cette partie, nous nous focalisons sur les mécanismes intervenant dans la couche Liaison de la norme 802.11. Ces outils permettent principalement de partager les ressources radio entre les différents équipements du réseau (multiplexage). Deux méthodes d'accès sont définies dans les spécifications de la norme WiFi : les protocoles DCF (*Distributed Coordination Function*) et PCF (*Point Coordination Function*). La couche Liaison réalise également le chiffrement des données par le biais du protocole WEP (*Wireless Encryption Protocol*). Cette opération permet de garantir la totale confidentialité des informations transmises. Nous considérons cependant que les données ne sont pas cryptées dans cette étude.

Description technique

Le DCF Ce mode représente le mécanisme de partage par défaut du support physique. Il est utilisé à la fois dans les réseaux *ad hoc* et infrastructure. Son implémentation est obligatoire dans les équipements WiFi. Cette fonction de coordination distribuée correspond à une méthode d'accès aléatoire, basée sur une version améliorée du CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*).

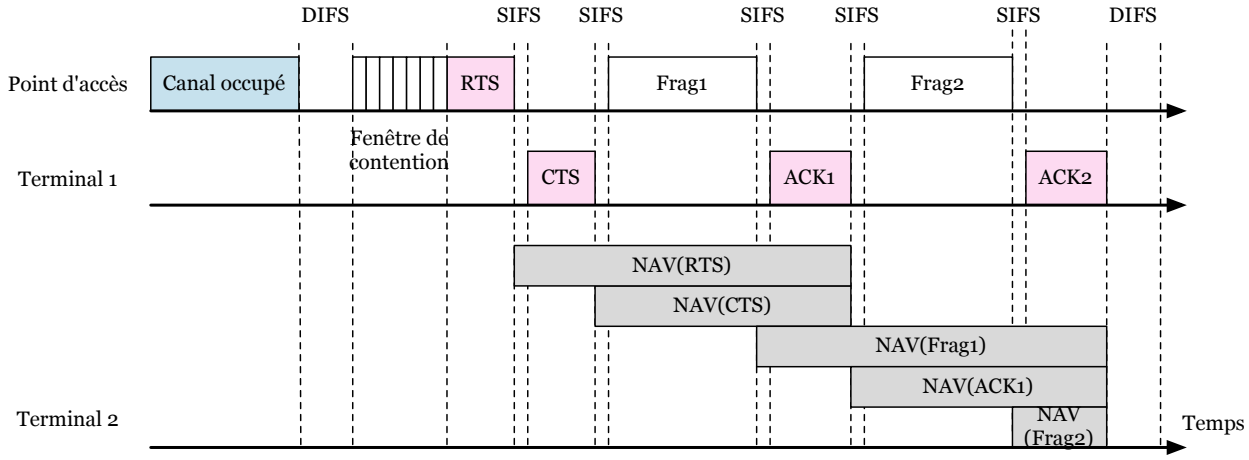


FIG. 2.15 – Protocole de transmission DCF de la couche MAC 802.11

Comme dans CSMA, une station souhaitant transmettre des données commence par attendre pendant une durée pré-définie (*DCF Inter Frame Space* ou DIFS de 50 μs). Si la couche Physique n'a détecté aucun signal sur le canal durant cette période, elle transmet un CCA à la couche MAC et la station patiente à nouveau pendant une durée aléatoire (fenêtre de contention). Si aucun signal n'a été détecté durant cet intervalle de temps aléatoire, la station envoie au récepteur une demande de réservation (*Request To Send* ou RTS). Cette courte trame de contrôle contient les adresses MAC de l'émetteur et du destinataire. Elle fournit également un champ d'information (*Duration*) contenant une estimation de la durée de transmission du premier fragment MAC. Cet indicateur permet aux autres stations de fixer leur NAV(RTS) (*Network Allocation Vector*). Durant un NAV, les stations adjacentes ne peuvent pas transmettre de données (pour éviter les collisions). Après un bref délai (*Short Inter Frame Space* ou SIFS de 10 μs), le récepteur renvoie une trame de contrôle CTS (*Clear To Send*) pour indiquer à l'émetteur qu'il accepte la transmission. Cette trame contient également une information temporelle permettant d'ajuster le NAV(CTS). La procédure de réservation du canal étant achevée, l'émetteur peut alors transmettre l'ensemble des données relatif à un paquet IP. Si la taille du paquet IP est trop importante, les données peuvent être réparties en plusieurs fragments MAC qui seront transmis successivement sur le canal. Le récepteur doit acquitter chaque fragment reçu en envoyant une trame de contrôle ACK (*Acknowledgment*). Lorsque l'émetteur ne reçoit pas l'acquittement d'un fragment, il retransmet automatiquement le fragment correspondant. Toutes les trames contiennent un champ d'information spécifiant la durée de transmission estimée du prochain fragment. Ce champ

permet de réajuster en permanence le NAV des stations voisines. Pour éviter les recouvrements temporels, un intervalle SIFS sépare chaque trame. A la fin de la transmission, la procédure de réservation du canal recommence et un autre utilisateur peut transmettre des données à son tour. La procédure de transmission DCF est illustrée sur la figure 2.15. Dans ce schéma, la transmission s'effectue entre le point d'accès et le terminal 1 et les données du paquet IP à transmettre ont été réparties en deux fragments MAC.

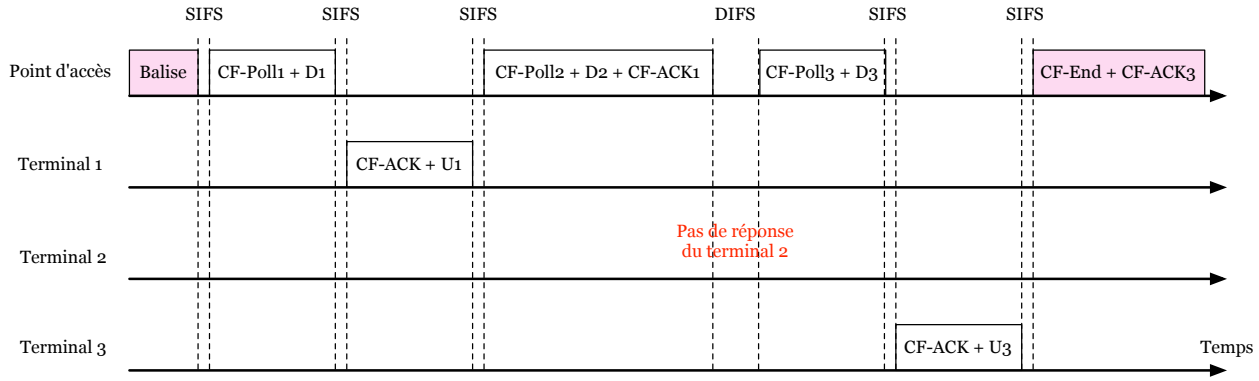


FIG. 2.16 – Protocole de transmission PCF de la couche MAC 802.11

Le PCF La méthode DCF est non déterministe et une station ne peut pas prévoir quand elle pourra transmettre des données. L'utilisation du mode PCF permet de pallier à cet inconvénient en fournissant une technique de transmission synchrone. Dans la méthode PCF, le point d'accès centralise les informations véhiculées sur le réseau (mode infrastructure) et supervise l'accès au support de chaque station. Pour éviter les collisions, le coordinateur interroge successivement chacune des stations en émettant une requête particulière (CF-Poll). Si le point d'accès veut transmettre des données à la station courante, il insère les données dans la trame de réservation (CF-Poll + D). Lorsque la station courante veut envoyer des données sur le réseau, elle répond au point d'accès en transmettant une trame d'acquittement contentant simultanément les données (CF-ACK + U). Durant cette période, les autres stations restent silencieuses et attendent leur tour. Une fois les informations transmises, le point d'accès interroge la station suivante tout en acquittant les données envoyées par la station précédente (CF-Poll + CF-ACK ou CF-Poll + CF-ACK + D). Si une station interrogée ne répond pas au point d'accès dans un temps imparti (*PCF Inter Frame Space* ou PIFS de 30 μs), ce dernier considère que la station ne souhaite pas transmettre d'information. Il passe alors à la station suivante. Un schéma récapitulatif de la méthode PCF est illustré sur la figure 2.16. Dans cet exemple, le point d'accès gère trois terminaux. Une communication en mode PCF est initialisé par la transmission d'une trame balise et se termine par l'envoi d'une trame CF-End ou CF-End + CF-ACK.

Le mode PCF est optionnel dans la norme 802.11 et il est toujours utilisé en alternance avec la méthode d'accès principale. La coexistence DCF/PCF est donc nécessaire. Pour réaliser cette fonctionnalité, la station coordinatrice alterne entre deux périodes : le mode

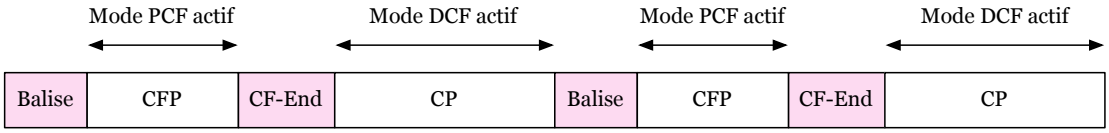


FIG. 2.17 – Gestion du temps par le point d'accès : alternance entre les modes PCF et DCF

PCF est activé durant une période CFP (*Contention Free Period*), suivi du mode DCF pendant une phase CP (*Contention Period*). Cette répartition temporelle est représentée sur la figure 2.17. Le point d'accès génère une trame balise pour indiquer le passage en mode PCF. Ce mode s'achève par la transmission d'une trame CF-End. Afin de gérer correctement ces deux modes, un PIFS est plus court qu'un DIFS. Une station n'intégrant pas le mécanisme PCF ne pourra donc pas émettre pendant la période CFP. Elle est ainsi obligée d'attendre la période de contention (CP).

En pratique, le mode PCF n'est jamais utilisé car les constructeurs ne l'intègre jamais dans leurs produits. Dans notre étude, nous considérons donc que le point d'accès et les terminaux mobiles ne supportent que le mécanisme DCF.

Format des paquets MAC associés au protocole DCF

Le protocole DCF permet de distribuer successivement les paquets IP entre le point d'accès et les terminaux en fournissant une méthode d'accès aléatoire au support hertzien. La couche Physique se charge ensuite d'insérer chaque paquet MAC dans le champ de contenu d'un paquet PHY. La procédure de réservation du canal s'effectue par un échange de trames RTS et CTS entre l'émetteur et le récepteur. Les fragments MAC incorporant les données du paquet IP sont ensuite transmis successivement au récepteur qui acquitte chaque fragment séparément. Dans cette partie, nous définissons le format des paquets intervenant lors de la transmission entre le point d'accès vers un terminal particulier. Pour faciliter la compréhension, nous commençons par spécifier le format des fragments de données avant de nous intéresser aux paquets de contrôle (RTS, CTS et ACK).

Les fragments MAC contiennent les données binaires du paquet IP segmenté. Ces données utiles sont encapsulées dans un champ de contenu (*payload*). Un en-tête (*header*) est ajouté au début de chaque fragment. Il contient les paramètres nécessaires à la couche Liaison du récepteur pour traiter les données. Par ailleurs, un code de détection d'erreurs est placé à la fin de chaque fragment et permet de valider l'intégrité des données reçues. Le format des fragments MAC est illustré sur la figure 2.18. Les champs d'information composant le fragment sont brièvement décrits ci-dessous :

- Les 2 bits du champ *Protocol Version* spécifient la version de la norme WiFi. Sa valeur est égale à 2 pour le standard 802.11.
- Les champs *Type* et *Subtype*, de 2 et 4 bits respectivement, identifient la fonction du

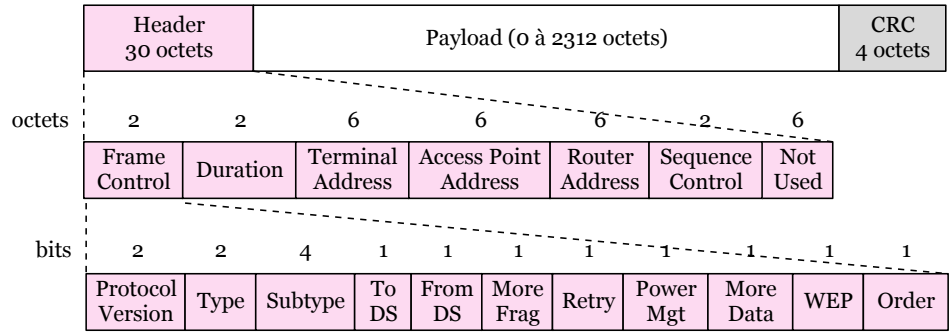


FIG. 2.18 – Format des fragments MAC

paquet. Pour un fragment de données en mode DCF, *Type* est fixé à 10_2 et *Subtype* à 0_{16}.

- Les deux champs *To DS* et *From DS* ont une longueur de 1 bit. Ces paramètres caractérisent l'architecture du réseau et la direction de transmission. Dans notre étude, nous nous intéressons à une transmission dans le sens descendant sur un réseau étendu : les données sont issues d'Internet et sont distribuées par le point d'accès vers les terminaux. *To DS* est fixé à 0 et *From DS* à 1.
- Le champ *More Frag* de 1 bit indique si le fragment courant représente le dernier fragment d'un paquet IP. Sa valeur vaut 1 si d'autres fragments suivent, 0 dans le cas contraire.
- Le champ *Retry* de 1 bit spécifie si le fragment courant représente une retransmission d'un fragment précédemment envoyé. Il est fixé à 1 si le fragment courant est retransmis et à 0 dans le cas contraire.
- Le champ *Power Mgt* de 1 bit spécifie si l'émetteur du fragment est en mode économie d'énergie. Nous ne considérons pas les mécanismes de gestion d'énergie dans cette étude et *Power Mgt* est fixé à 0.
- Le champ *More Data* de 1 bit est utile lorsque le protocole PCF est activé. Il indique à une station en mode économie d'énergie si le point d'accès possède d'autres informations à lui transmettre. Suivant les hypothèses énoncées ci-dessus, *More Data* est fixé à 0.
- Le champ *WEP* de 1 bit spécifie si le contenu de données du fragment courant est crypté. Dans notre cas, nous négligeons les mécanismes de chiffrement et *WEP* est égal à 0.
- Le champ *Order* de 1 bit indique si les fragments sont transmis dans l'ordre. Dans notre étude, la transmission est ordonnée et *Order* vaut 1.
- Le champ *Duration* contient une valeur sur 2 octets qui spécifie la durée de transmission estimée (en microsecondes) du futur fragment (voir figure 2.15). Ce paramètre considère aussi l'ensemble des données protocolaires (SIFS, ACK) attachées au futur fragment. Cette valeur varie entre 0 et $2^{15} - 1$ dans le mode DCF et permet aux autres stations d'ajuster leur NAV. Elle est calculée en fonction de la taille du futur fragment, du débit utilisé pour transmettre le fragment courant.
- Les 6 octets du champ *Terminal Address* représentent l'adresse MAC du récepteur.

- Les 6 octets du champ *Access Point Address* correspondent à l'adresse MAC du point d'accès.
- Les 6 octets du champ *Router Address* représentent l'adresse MAC du routeur local connecté à internet.
- Le champ *Sequence Control* de 2 octets contient deux paramètres : un numéro de séquence et un numéro de fragment. Le numéro de séquence est codé sur 12 bits et sa valeur représente le compteur du paquet IP courant. Le numéro de fragment est codé sur 4 bits et identifie le compteur du fragment MAC courant. Ces deux paramètres permettent de réassembler les fragments reçus pour reconstruire les paquets IP dans le terminal.
- Le dernier champ de l'en-tête est composé de 6 octets. Il est réservé pour les transmissions sur les réseaux locaux. Dans notre étude, tous les bits sont fixés à 0.
- La *payload* contient les données utiles à transmettre : elle représente les données appartenant à une portion d'un paquet IP. Ce champ peut contenir de 0 à 2312 octets.
- Un CRC de 4 octets protège les champs du *header* et les données contenues dans la *payload*. Son polynôme générateur est défini par : $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$. Il permet au récepteur de détecter des fragments erronés et de demander leur retransmission.

Contrairement aux fragments de données, les paquets de contrôle MAC ne contiennent pas de *payload*. Il sont juste composés d'un en-tête et d'un CRC. Nous définissons ci-dessous le format des trames RTS, CTS et ACK.

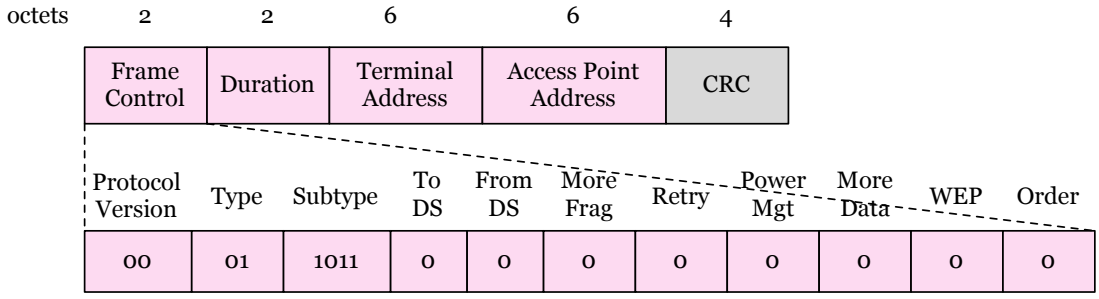


FIG. 2.19 – Format d'une trame RTS

Un paquet de contrôle RTS est transmis par le point d'accès quand il souhaite envoyer des informations à un terminal spécifique. Il contient donc l'adresse MAC du point d'accès ainsi que l'adresse MAC du terminal cible. Le format complet d'une trame RTS est exposé sur la figure 2.19.

Le terminal répond au point d'accès en émettant une trame CTS. Ce paquet est plus court qu'une trame RTS puisqu'il ne contient que l'adresse du point d'accès. Son format est illustré sur la figure 2.20.

Quand le terminal reçoit correctement un fragment de données, il transmet une trame ACK au point d'accès. Ce paquet d'acquittement possède la même taille qu'une trame CTS.

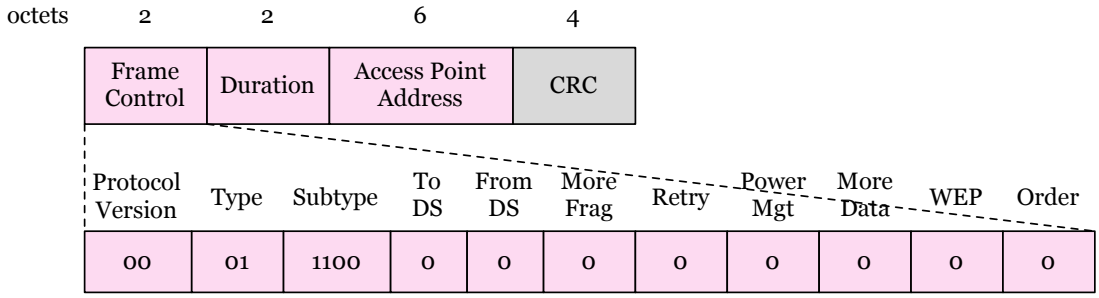


FIG. 2.20 – Format d'une trame CTS

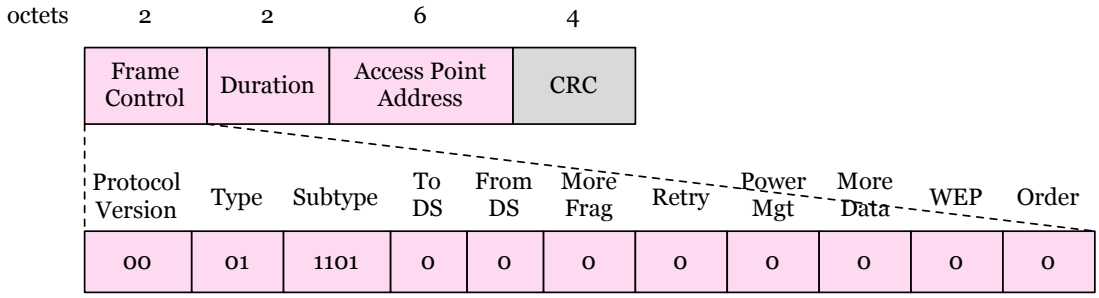


FIG. 2.21 – Format d'une trame ACK

Son format est représenté sur la figure 2.21.

Remarque 2 *En réalité, la couche Liaison est constituée d'un empilement de deux couches : la couche MAC (dont le protocole est adapté au support physique) et la couche LLC (*Logical Link Control*) [37], respectant le standard* 802.2*. La couche LLC permet d'établir le lien logique entre la couche MAC et la couche Réseau. Lorsque la couche MAC* 802.11 *est utilisée, la couche LLC spécifie juste le protocole employé à la couche supérieure. Actuellement, la majorité des transmissions repose sur le protocole IP et la couche LLC devient progressivement obsolète avec le développement croissant d'Internet. Dans ce travail, nous considérons que les informations fournies par cette couche sont redondantes et nous la négligeons pour simplifier notre étude. Dans une situation concrète, une solution efficace pour aiguiller les paquets vers les services appropriés de la couche Réseau consisterait à rajouter un champ complémentaire dans l'en-tête des fragments MAC. Ce principe est déjà utilisé dans la norme Ethernet grâce au champ* EtherType*.*

2.3.3 La couche Réseau (IP)

La fonction principale de la couche IP consiste à acheminer les informations de l'émetteur vers le récepteur. Cette tâche est réalisée par les routeurs qui utilisent une table de routage

pour rapprocher les paquets de leur destination finale. La couche IP permet également de fragmenter les paquets. Cette opération est cependant désactivée dans ce travail pour simplifier les simulations. Bien que le protocole IP ne soit pas fiable, nous considérons que toutes les données transmises arrivent au point d'accès (nous négligeons les problèmes associés au réseau coeur : pas de congestion ou d'égarement). Par contre, l'ordre des paquets n'est pas garanti à la réception. Il existe deux protocoles IP : IPv4 et IPv6. Dans cette étude, nous utilisons le protocole IPv4 qui est actuellement le plus répandu.

Format des paquets IP

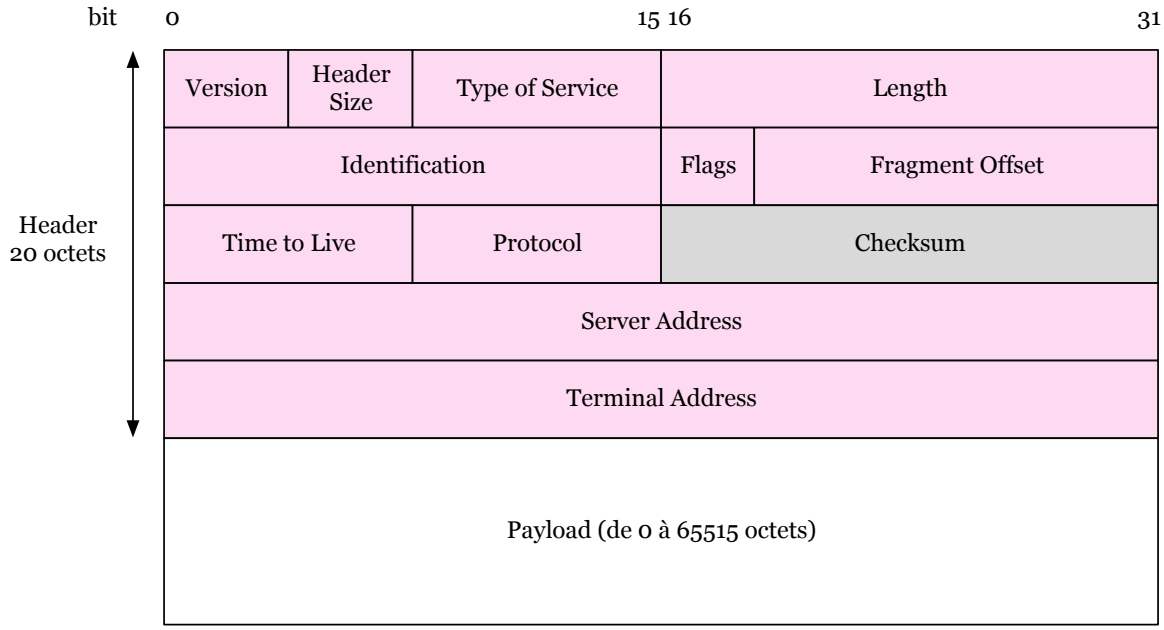


FIG. 2.22 – Format des paquets IP

Les paquets IP contiennent les informations binaires des paquets UDP. Ces données utiles sont encapsulées dans un champ de contenu (*payload*). Un en-tête (*header*) est ajouté au début de chaque paquet. Il contient les paramètres nécessaires à la couche Réseau pour traiter les données. Le format des paquets IP est illustré sur la figure 2.22. Les champs d'information composant le paquet sont spécifiés ci-dessous :

- Le champ *Version* de 4 bits spécifie la version du protocole IP. Sa valeur est fixée à 4 pour le protocole IPv4.
- Le champ *Header Size* de 4 bits spécifie la taille de l'en-tête en mots de 32 bits. Dans notre cas, sa valeur vaut 5.
- Le champ *Type of Service* de 1 octet spécifie les contraintes de la transmission (débit, délai, fiabilité) en fonction des données transportées. Dans cette étude, nous utilisons le mode par défaut et la valeur est fixée à 00_{16}.

- Les 2 octets du champ *Length* indiquent la taille total du paquet IP en octets. Cette valeur tient compte de l'en-tête et varie entre 20 et 65535.
- Le 2 octets du champ *Identification* représentent un identifiant unique caractérisant le paquet IP. Ce paramètre est généré aléatoirement et sa valeur change dans chaque paquet.
- Les champs *Flags* et *Fragment Offset*, de 3 et 13 bits respectivement, contiennent les paramètres associés à la fragmentation. Dans notre étude, la valeur globale, codée sur 2 octets, est égale à 4000_{16}.
- Le champ *Time to Live* de 1 octet représente le nombre maximal de routeurs que le paquet IP peut traverser. Au niveau du serveur, cette valeur est généralement initialisée à 255. Chaque routeur décrémente ce paramètre de 1. Lorsque la valeur atteint 0, le paquet est effacé. Cette mesure de sécurité permet de détruire les paquets perdus dans le réseau.
- Le champ *Protocol* de 1 octet spécifie le protocole de la couche supérieure. UDP correspond à la valeur 17.
- Un checksum de 2 octets protège les paramètres de l'en-tête. Il permet aux routeurs ou au terminal de détecter un en-tête erroné. Quand une erreur est détectée, le paquet complet est effacé.
- Les 4 octets du champ *Server Address* représentent l'adresse IP du serveur.
- Les 4 octets du champ *Terminal Address* représentent l'adresse IP du terminal.
- La *payload* encapsule les données utiles à transmettre (paquet UDP). Ce champ peut contenir entre 0 à 65515 octets.

2.3.4 La couche Transport (UDP/RTP)

Les protocoles de la couche Transport interviennent directement entre le serveur et le terminal. En définitive, cette couche gère les communications de bout en bout entre processus. Le protocole TCP est utilisé à ce niveau dans la majorité des systèmes. Ce protocole fournit un service orienté connexion qui garantit la fiabilité du transport. Il se charge notamment de retransmettre les paquets TCP perdus ou dégradés et délivre un flux ordonné à la couche supérieure. Dans notre situation, le délai engendré par les procédures du mode connecté ne peut pas être toléré. Il est alors nécessaire d'utiliser un protocole plus souple s'appuyant sur une transmission sans connexion (non-fiable) : le protocole UDP. Ce protocole est cependant relativement simple et doit souvent être associé à un protocole plus complet pour réaliser une jonction efficace entre le réseau et l'application multimédia. Il s'agit du protocole RTP. Dans cette partie, nous présentons brièvement ces deux protocoles.

Format des paquets UDP

Le protocole UDP gère le multiplexage des paquets de manière très simple en utilisant le concept des ports. Les ports permettent aux équipements de différencier les utilisateurs et les processus. Le protocole UDP travaille en mode non-connecté et ne garantit pas la fiabilité

de la communication. En revanche, il est capable de détecter les paquets corrompus.

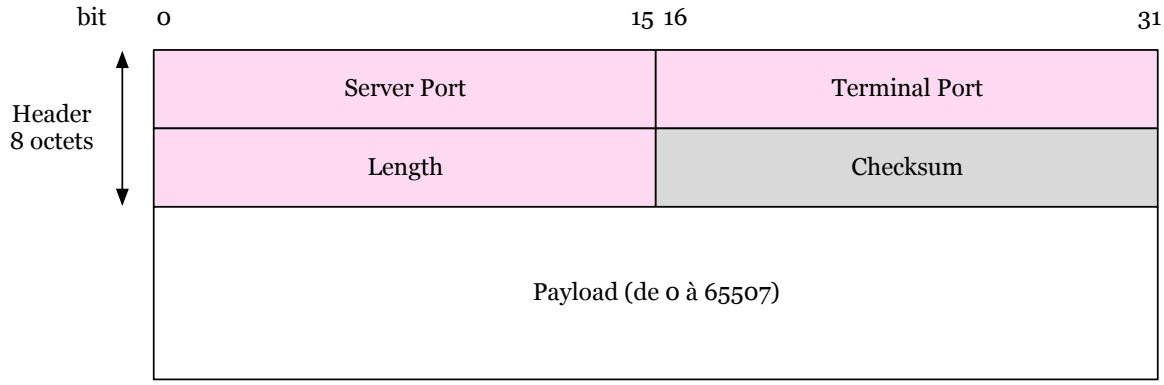


FIG. 2.23 – Format des paquets UDP

Les paquets UDP encapsulent les données des paquets RTP. Ces informations sont placées dans un champ de contenu (*payload*). Un en-tête (*header*) est ajouté au début de chaque paquet et contient les paramètres utilisés par la couche UDP. Le format des paquets UDP est exposé sur la figure 2.23 et les champs d'information sont définis ci-dessous :

- Les 2 octets du champ *Server Port* représentent le port d'émission du serveur. Si le terminal veut répondre au serveur, il doit utiliser ce port.
- Les 2 octets du champ *Terminal Port* correspondent au port de réception du terminal.
- Les 2 octets du champ *Length* spécifient la taille en octets du paquet UDP. Cette taille tient compte de l'en-tête. Sa valeur varie entre 0 et 65515 (en prenant en compte l'en-tête de la couche IP).
- Un checksum de 2 octets protège l'ensemble des données du paquet. Il permet au terminal de détecter un paquet contenant des erreurs. Quand une erreur est détectée, le paquet est effacé.
- La *payload* regroupe les données utiles à transmettre (paquet RTP). Ce champ peut contenir entre 0 à 65507 octets (en tenant compte de la taille des en-têtes IP et UDP).

Format des paquets RTP

Le protocole RTP fournit les services requis pour transmettre efficacement des données IP soumises à des contraintes temps-réel. Ce protocole est directement associé à l'application courante et certains paramètres RTP sont même utilisés à la couche supérieure. Il permet notamment de réordonner les paquets reçus, de contrôler l'arrivée à destination des paquets, de synchroniser les flux décodés (marqueurs temporels) et d'identifier le type des informations transportées.

Les paquets RTP regroupent les informations générées par la couche Application. Dans notre cas, elle va encaspuler les paquets NAL générés par le codeur H.264/AVC. Ces données

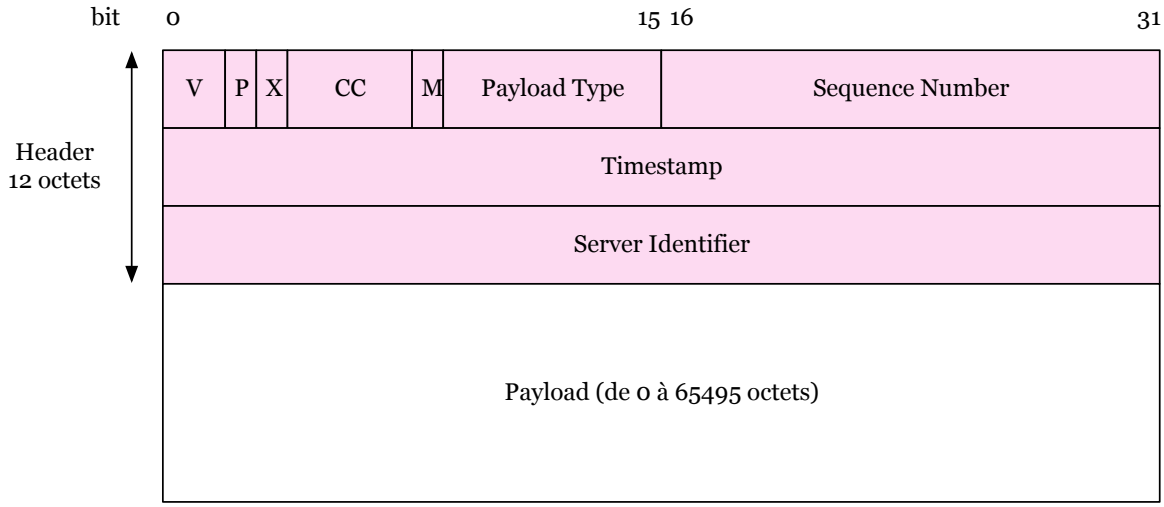


FIG. 2.24 – Format des paquets RTP

sont placées dans un champ de contenu (*payload*). Un en-tête (*header*) est ajouté au début de chaque paquet et contient les paramètres utiles de la couche RTP. Le format des paquets RTP est illustré sur la figure 2.24 et les champs d'information sont décrits ci-dessous :

- Les 2 bits du champ *V* ou *Version* spécifient la version du protocole RTP. Cette valeur est toujours égale à 2.
- Le champ *P* ou *Padding* de 1 bit précise si des octets de bourrage sont ajoutés à la fin du paquet. Dans notre cas, ce paramètre vaut 0.
- Le champ *X* ou *Extension* indique si des extensions sont ajoutées dans l'en-tête du paquet. Dans notre étude, ce paramètre vaut 0.
- Lorsque plusieurs utilisateurs interagissent entre eux (audioconférence), un identifiant CSRC caractérise chaque source (utilisateur). Chaque utilisateur envoie ses informations à un point central (*mixer*), qui se charge de les mélanger et de retransmettre les données fusionnées aux différents utilisateurs. Chaque paquet RTP retransmis par le point central contient l'ensemble des sources contributrices (CSRC). Ces identifiants sont ajoutés à la suite de l'en-tête. Les 4 bits du champ *CC* ou *CSRC Count* spécifient le nombre d'identifiants CSRC. Dans cette étude, la communication se limite à un serveur et un terminal et la valeur de *CC* est fixée à 0.
- Le champ *M* ou *Marker* de 1 bit est utilisé par le décodeur. Il permet de séparer les images : il est fixé à 1 dans un paquet RTP contenant le dernier *slice* d'une image, à 0 pour les *slices* intermédiaires.
- Les 7 bits du champ *Payload Type* identifient le contenu des données transportées dans le paquet RTP. Pour la norme H.264/AVC, ce paramètre vaut 105.
- Les 2 octets du champ *Sequence Number* représentent le compteur du paquet RTP. Sa valeur est incrémentée de 1 pour chaque paquet transmis. Ce paramètre est utilisé pour réordonner les paquets RTP à la réception.
- Les 4 octets du champ *Timestamp* déterminent l'instant où l'image doit être affichée.

Ce paramètre est utilisé par le décodeur pour synchroniser l'affichage des images. Pour la norme H.264, sa valeur dépend des cycles d'une horloge cadencée à 90 kHz. Dans la première image d'un flux vidéo, ce champ est initialisé à 0. Ce paramètre est constant dans les paquets RTP contenant les *slices* d'une même image.

– Les 4 octets du champ *Server Identifier* correspond à un unique identifiant caractérisant le serveur.
– La *payload* regroupe les données utiles à transmettre (paquet NAL). Ce champ peut contenir entre 0 à 65495 octets (en tenant compte de la taille des en-têtes IP, UDP et RTP).

Remarque 3 *La couche Application (APL) contient un encodeur (côté serveur) ou un décodeur (côté terminal) respectant la norme H.264/AVC. Le principe de fonctionnement de ces éléments a largement été expliqué dans le chapitre 1 et il est inutile d'y revenir. Retenons juste que le flux vidéo encodé est réparti dans des paquets NAL qui sont décrits dans la partie 1.3.3.*

Chapitre 3

Transmission vidéo robuste

Dans ce chapitre, nous introduisons le premier travail abordé durant cette thèse. Une introduction approfondie permet de faire le lien avec les deux premiers chapitres et de cibler les problèmes intervenant lors d'une transmission vidéo.

3.1 Position du problème

Comme nous l'avons vu au chapitre 1, le processus de compression vidéo s'appuie fortement sur l'exploitation des redondances temporelles entre les images (prédiction Inter). Dans un second temps, tous les symboles du flux sont remplacés par des mots de code de longueur variable par le biais du codeur entropique. Chaque image représente donc une succession de mots de code VLC, respectant une structure syntaxique et sémantique fournie par la norme, et encapsulés dans des paquets NAL. Les images sont rassemblées pour former des séquences (GOP) et chaque GOP débute par une image I qui sert de point d'ancrage au décodeur. Les images d'un GOP sont comprimées puis transmises successivement sur le réseau.

Dans le cas d'une transmission analogique, les paquets de données arrivant à l'entrée du décodeur source peuvent contenir des erreurs. Lorsque le décodeur traite le flux binaire et rencontre une simple erreur de transmission, il décode potentiellement un symbole erroné. D'après la condition du préfixe, le mot de code VLC associé au symbole décodé possède une longueur binaire différente du mot de code original. Cette erreur entraîne donc une désynchronisation complète du décodeur sur le flux vidéo et le reste de l'image est perdu (les blocs suivants ne peuvent pas être décodés et l'information visuelle de ces blocs est perdue). Ce principe de désynchronisation est illustré sur la figure 3.1. Le décodeur doit alors attendre le début de l'image suivante pour se resynchroniser sur le flux (en utilisant les séquences de synchronisation séparant les paquets). Néanmoins, cette image dégradée sert ensuite de référence pour construire le signal de prédiction des images suivantes. L'erreur visuelle se propage donc à travers les images de manière croissante et désordonnée. Ce phénomène peut conduire à une destruction complète d'une séquence d'images. Seule l'arrivée d'un nouveau

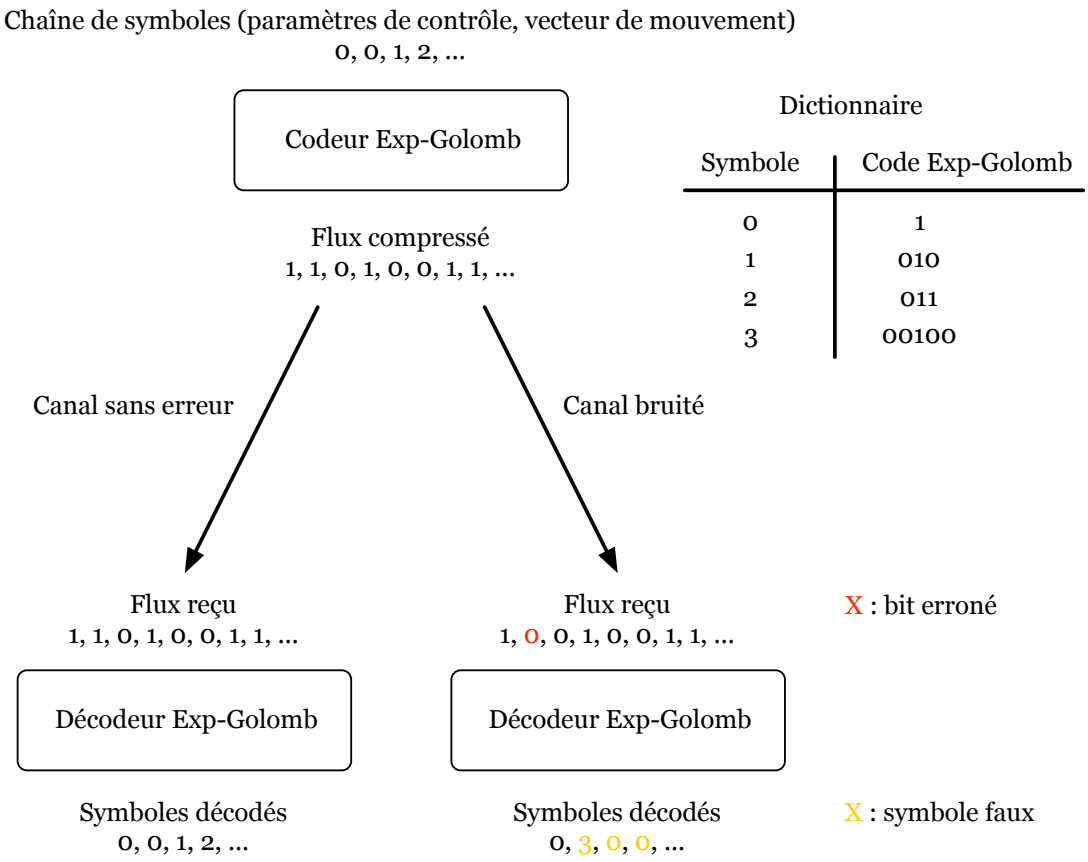


FIG. 3.1 – Désynchronisation du décodeur liée à une erreur de transmission

GOP permet au décodeur de retrouver un flux vidéo cohérent. Le processus de propagation d'erreurs est illustré sur la figure 3.2. Par conséquent, une simple erreur binaire peut provoquer des dégâts considérables sur la vidéo décodée. Nous aboutissons donc à la constatation suivante : plus une vidéo est comprimée, plus elle est sensible aux erreurs de transmission. Pour garantir un décodage efficace, le flux arrivant au niveau du décodeur doit être totalement exempt d'erreur.

Dans les transmissions radio modernes, le signal reçu peut être fortement dégradé et n'est pas directement utilisable par le décodeur. Pour résoudre ce problème, les protocoles actuels répartissent les informations d'une image dans plusieurs fragments. Pour assurer la transmission des fragments entre l'émetteur et le récepteur, le système de transmission est basé sur une pile protocolaire contenant un ensemble de couches. Chaque couche réalise une fonction spécifique et ajoute des paramètres de contrôle aux informations à transmettre. Ces paramètres sont utilisés dans les équipements intermédiaires participant au routage des paquets. Ces notions ont largement été abordées dans le chapitre 2. Dans notre situation, nous nous focalisons sur le transfert de vidéo entre un serveur relié à Internet et un terminal mobile rattaché à un point d'accès de type WiFi. Nous considérons que le réseau coeur est idéal et nous portons exclusivement notre attention sur les problématiques associées à la

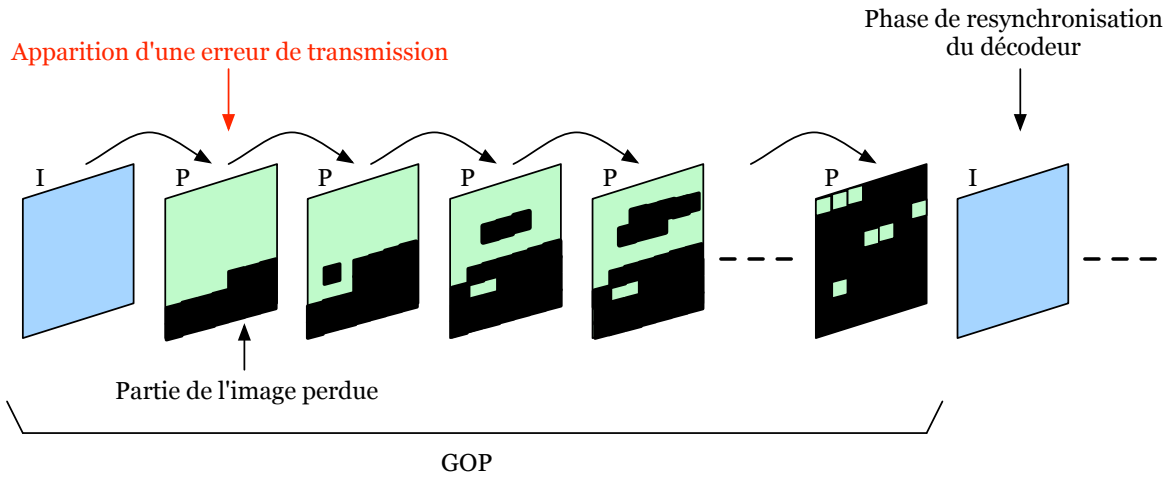


FIG. 3.2 – Phénomène de propagation des erreurs entre les images

transmission radio.

Au niveau du récepteur, la restitution des données suit la procédure exposée sur la figure 3.3. Les informations reçues par la couche PHY sont traitées à l'aide des techniques introduites dans la partie 2.3.1. Les données utiles sont ensuite transmises à la couche MAC (voir partie 2.3.2) qui vérifie leur intégrité par le biais d'un CRC. Lorsqu'une erreur est détectée, le fragment est automatiquement retransmis par le point d'accès. Quand tous les fragments ont été correctement reçus par le récepteur, la couche MAC les assemble pour former un paquet IP. Après avoir analysé les paramètres contenus dans l'en-tête du paquet, la couche IP (voir partie 2.3.3) transmet son contenu à la couche supérieure. A ce niveau (voir partie 2.3.4), la couche UDP vérifie l'intégrité de la totalité des données du paquet par le biais d'un checksum. Le protocole UDP n'étant pas un mode orienté connexion, les paquets UDP contenant des erreurs sont effacés. Les autres paquets remontent ensuite à la couche RTP, puis atteignent finalement la couche APL où ils sont traités par le décodeur source.

Nous constatons donc que de nombreux mécanismes de protection (correction/détection d'erreurs) sont répétés au niveau de chaque couche protocolaire. En théorie, ces outils garantissent que tous les paquets vidéo transmis seront correctement reçus par la couche APL du terminal. Cependant, lorsque les conditions de transmission se dégradent significativement, les fragments arrivant à la couche MAC peuvent être constamment erronés. Pour éviter qu'un fragment ne soit retransmis en permanence et sature le réseau, le protocole de la couche MAC limite le nombre de retransmissions à une valeur seuil (généralement fixée à 10). Lorsque le nombre de tentatives atteint ce seuil, le fragment est effacé. Les conséquences sont alors dramatiques : le paquet IP (associé au fragment effacé) est perdu, le paquet vidéo n'atteint pas le décodeur et l'image complète est perdue. Ces situations ne sont pas rares dans les transmissions radio-mobiles car le canal subit des dégradations prolongées, pouvant parfois atteindre quelques secondes. Ce phénomène illustre la problématique des transmissions en mode paquet. Dans un tel système, le décodeur doit être capable de faire face à des pertes

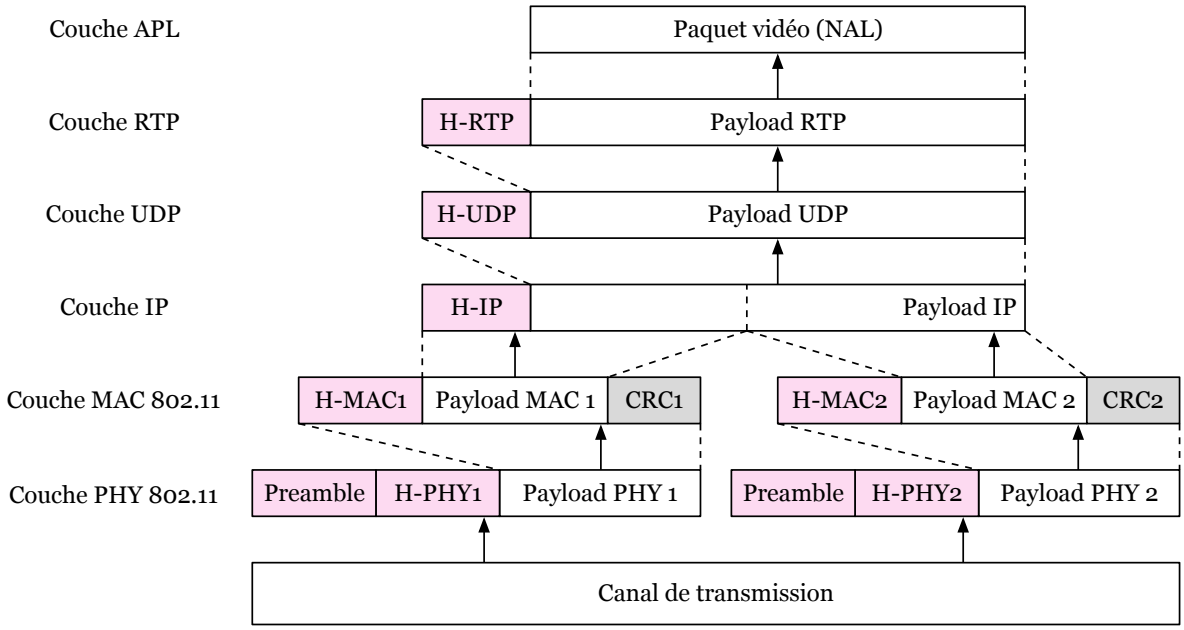


FIG. 3.3 – Pile protocolaire du terminal WiFi

de paquets vidéo.

Pour palier ce problème, une solution basique consiste à intégrer un codeur/décodeur canal (de type Reed-Solomon) à des couches intermédiaires pour lutter contre l'effacement des paquets [38, 39]. Dans cette technique, l'émetteur génère des bits de redondance associés à chaque paquet et les répartit sur un ensemble de paquets. A la réception, un paquet perdu peut être reconstruit en utilisant les redondances contenues dans les autres paquets. Cette méthode, bien que très efficace, possède deux inconvénients majeurs :

1. Cet outil est souvent implémenté dans la couche RTP. Le codeur intervient donc dans le serveur et ne connaît pas précisément la qualité instantanée du canal radio. Par conséquent, la quantité de redondances est souvent mal dimensionnée. Elle peut être excessive quand le canal est faiblement perturbé, réduisant la bande-passante allouée pour les autres informations. Dans la situation inverse, elle peut être insuffisante lorsque les conditions se dégradent et conduire à la perte du paquet.
2. Pour reconstruire un paquet, le décodeur canal doit attendre que tous les paquets complémentaires soient arrivés au récepteur. Ce procédé entraîne donc une augmentation du délai de transmission globale. Une durée trop importante peut devenir gênante car elle intervient directement sur le temps de chargement de la vidéo.

Une autre solution intervient au niveau du codeur/décodeur source. Dans la partie 1.3.4, nous avons étudié deux mécanismes : le FMO et l'ASO. La combinaison de ces techniques permet à l'encodeur H.264/AVC de diviser les images en plusieurs régions entrelacées (*slices*) et d'encoder les *slices* de manière indépendante : les macroblocs d'un *slice* peuvent être

décodés sans utiliser les informations contenues dans un autre *slice*. Par ailleurs, les *slices* encodés d'une image sont transmis dans des paquets séparés. Au niveau du récepteur, la perte d'un paquet aura un impact limité sur la qualité vidéo. Seule la zone visuelle correspondant aux informations perdues ne pourra pas être reconstruite. Les autres *slices*, contenus dans les paquets correctement reçus, pourront être totalement décodés. Ce scénario est illustré sur la figure 3.4. Dans cet exemple, l'image transmise est découpée en deux *slices* entrelacés.

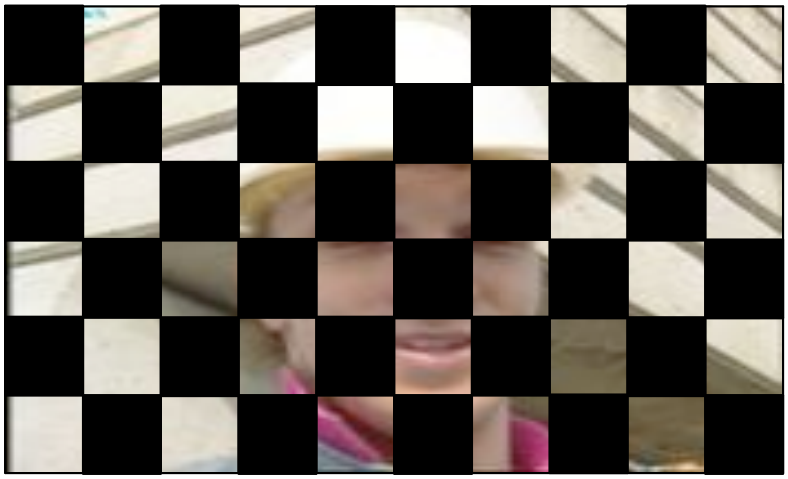


FIG. 3.4 – Conséquence de la perte d'un *slice* au niveau du décodeur

Une idée intuitive consiste à estimer les blocs manquants des images en utilisant les informations contenues dans les zones correctement décodées. Dans cette technique, le décodeur cherche donc à dissimuler les erreurs (*error concealment*) en générant une représentation des blocs perdus qui soit la mieux adaptée au reste des informations de l'image. Ces méthodes sont basées sur l'exploitation des redondances temporelles et spatiales entre les *slices*. Elles reposent sur les hypothèses suivantes :

- Le contenu des images varie de manière continue dans le domaine spatial : les images sont faiblement texturées.
- Les variations temporelles sont continues : le déplacement des objets à travers les images reste régulier ou évolue lentement (pas de changement de scène abrupte).

De telles techniques sont basées sur des outils d'interpolation : ils ne garantissent pas une solution optimale. Par conséquent, les erreurs de prédiction se propagent entre les images jusqu'à l'arrivée d'une image I. En pratique, ces mécanismes sont largement employés car ils fournissent des résultats visuels intéressants pour une complexité souvent réduite. Par ailleurs, aucune information complémentaire ne doit être transmise au récepteur, le décodeur utilise juste les redondances visuelles inhérentes de la vidéo décodée. Il faut cependant noter que les performances atteintes par ces mécanismes dépendent fortement de la répartition des *slices*. Plus les *slices* de l'image sont entrelacés, plus les redondances spatiales et temporelles sont importantes. En contrepartie, l'entrelacement des *slices* réduit le taux de compression de

l'encodeur (étant donné que les *slices* sont indépendants). Il faut donc trouver un compromis entre compression et robustesse.

Il existe une quantité importante de documents proposant des techniques d'*error concealment*. En revanche, seuls quelques schémas simples sont utilisés dans les applications pratiques. A titre d'illustration, les travaux présentés dans [40, 41, 42] introduisent des méthodes tirant parti des redondances spatiales. Des techniques exploitant les redondances temporelles sont présentées dans [43, 44, 45, 46]. Des approches hybrides, basées sur l'exploitation conjointe des redondances spatiales et temporelles, peuvent être trouvées dans [47, 48, 49].

3.2 Les techniques de décodage conjoint source-canal

Ces dernières années, des techniques de décodage conjoint source-canal (JSCD) ont été proposées pour corriger les paquets vidéo corrompus. Ces méthodes impliquent des décodeurs source robustes, qui exploitent les redondances inhérentes des paquets reçus, pour corriger les erreurs de transmission. Plusieurs sources de redondance ont été identifiées. Les contraintes associées à la syntaxe des codes VLC [50, 51, 52, 53, 54] ont été utilisées en premier lieu. Puis, les propriétés liées à la sémantique du flux comprimé ont été combinées aux redondances syntaxiques pour améliorer les performances des décodeurs robustes [55, 56, 57, 58]. Les informations relatives à la paquétisation des données encodées ont été exploitées dans [59]. Récemment, les redondances introduites par le codeur canal ont été conjointement employées avec les redondances laissées par le codeur source dans un traitement de décodage itératif [60, 61, 62]. Ces schémas joints améliorent les performances de décodage en comparaison avec les méthodes traditionnelles.

Dans ce chapitre, nous présentons une méthode de décodage robuste exploitant conjointement les propriétés sémantiques et syntaxiques du flux vidéo encodé ainsi que les redondances fournies par le CRC de la couche MAC. Dans la technique proposée, le CRC n'est pas utilisé pour détecter les erreurs dans les fragments MAC, il est considéré comme un code de correction d'erreurs qui est utilisé pour améliorer les performances de décodage de la vidéo. Cette approche a déjà été introduite dans [63, 64]. Dans leurs travaux, le CRC est utilisé pour corriger des erreurs dans les fragments MAC. Dans cette étude, nous proposons de le combiner aux redondances laissées par le codeur source pour renforcer la qualité du décodage vidéo. Ce traitement est basé sur une variété de techniques (décodage souple de codes en bloc [65, 66], décodage séquentiel [67], décodage source exploitant les redondances sémantiques et syntaxiques du flux vidéo [57]) qui sont combinées pour atteindre notre objectif.

Par ailleurs, la méthode proposée doit être intégrée dans le décodeur source du récepteur. Il est donc nécessaire de modifier la pile protocolaire du terminal pour pouvoir disposer des informations souples relatives aux fragments MAC au niveau de la couche APL. Cette nouvelle architecture protocolaire, dans laquelle les couches PHY, MAC et APL travaillent de manière complémentaire, est présentée ci-dessous. Trois modifications sont nécessaires pour implémenter la solution proposée :

- La couche PHY doit posséder un décodeur canal SISO (*Soft-Input Soft-Output*) pour traiter les signaux captés par l'antenne. Les informations souples issues du décodage sont transmises à la couche supérieure.
- Au niveau de la couche MAC, la procédure de détection d'erreurs basée sur le CRC doit être désactivée. Nous verrons ultérieurement que les retransmissions peuvent être contrôlées par un critère dépendant de la qualité des informations souples (voir chapitre 4). A ce stade, retenons juste que le récepteur acquitte tous les paquets d'information. Les fragments complets (composés des informations souples relatives au *header*, à la *payload* et au CRC) sont ensuite concaténés pour reconstituer les paquets IP.
- Les couches IP, UDP et RTP intègrent des protocoles perméables qui laissent remonter l'ensemble des informations jusqu'à la couche APL.

Ces modifications peuvent être facilitées en utilisant les mécanismes décrits dans le chapitre 4, qui introduisent un nouveau modèle de couche perméable basée sur la correction des en-têtes protocolaires erronés. Ainsi, nous pouvons considérer, dans la suite de ce chapitre, que les en-têtes des paquets reçus sont corrects dans toutes les couches protocolaires.

En suivant ces modifications, la couche APL reçoit une succession de fragments MAC, contenant les informations souples fournies par la couche PHY. Ces transformations ne concernent que le récepteur (les opérations de l'émetteur et le signal transmis restent inchangés). Le nouveau format des paquets reçus par la couche APL est représenté sur la figure 3.5.

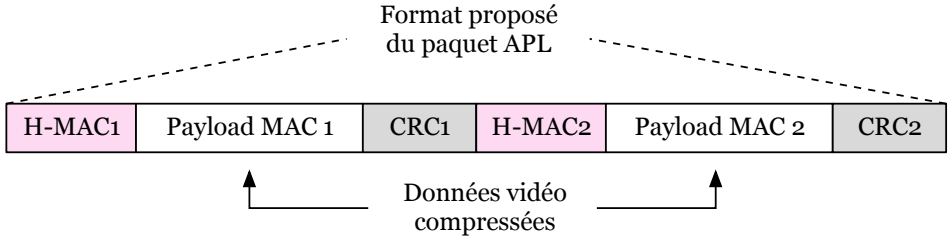


FIG. 3.5 – Format des nouveaux paquets APL

L'organisation de ce chapitre respecte le plan suivant : une technique générale de décodage séquentiel est proposée dans la partie 3.3. La partie 3.4 introduit une autre méthode de décodage simplifiée et optimisée. La partie 3.5 traite de la réduction de la complexité algorithmique. Finalement, les résultats de simulation sont fournis dans la partie 3.6.

3.3 Schéma général de décodage séquentiel

Durant l'opération d'encodage de la vidéo, le codeur source commence par générer les symboles associés aux différents paramètres de compression (vecteurs de mouvement, résidus

de prédiction, etc). Ces symboles sont ensuite convertis en mots binaires de longueur variable par le biais d'un codeur entropique. Ces deux opérations font intervenir des propriétés sémantiques et syntaxiques qui dépendent de la norme utilisée.

De manière théorique, considérons un encodeur H.264/AVC générant K symboles et les encapsulant dans un vecteur $\mathbf{m}$, tel que $\mathbf{m} = [m_1 \dots m_K]$. Le codeur entropique associe un mot de code VLC $\mathbf{x}_{m_i}$ de $\ell(\mathbf{x}_{m_i})$ bits à chaque symbole m_i, $i = 1 \dots K$. A la sortie du codeur, tous les symboles sont convertis en une séquence binaire $\mathbf{x}$ de $\ell(\mathbf{x})$ bits, tel que

$$\mathbf{x} = [\mathbf{x}_{m_1} \dots \mathbf{x}_{m_K}] \text{ avec } \sum_{i=1}^{K} \ell(\mathbf{x}_{m_i}) = \ell(\mathbf{x}).$$

Au niveau de la couche MAC, un en-tête $\mathbf{h}$ de $\ell(\mathbf{h})$ bits est concaténé au vecteur $\mathbf{x}$ et l'ensemble forme le vecteur $\mathbf{d} = [\mathbf{h}, \mathbf{x}]$ de $\ell(\mathbf{d}) = \ell(\mathbf{h}) + \ell(\mathbf{x})$ bits. Un CRC $\mathbf{c}$ de $\ell(\mathbf{c})$ bits est ensuite déterminé à partir du vecteur $\mathbf{d}$ et ajouté à la fin du paquet. Cet ensemble d'information est collecté dans le vecteur

$$\mathbf{t} = [\mathbf{h}, \mathbf{x}, \mathbf{c}] = [\mathbf{d}, \mathbf{c}],$$

où $\mathbf{c} = \mathcal{F}(\mathbf{d})$, $\mathcal{F}$ étant une fonction d'encodage générique.

L'évaluation de $\mathbf{c}$ dépend d'un polynôme générateur $g(z) = \sum_{i=0}^{\ell(\mathbf{c})} a_i z^i$ caractérisant le CRC [68]. Une matrice génératrice systématique $\mathbf{G} = [\mathbf{I}, \mathbf{\Pi}]$ peut être associée à $g(z)$, où $\mathbf{I}$ représente une matrice identité et $\mathbf{\Pi}$ correspond à une matrice de parité. Le CRC $\mathbf{c}$ peut alors être déterminé à partir de $\mathbf{G}$ en utilisant un traitement récursif sur les $\ell(\mathbf{d})$ bits du vecteur $\mathbf{d}$ comme suit

$$\mathbf{c}^{j+1} = \mathcal{F}(\mathbf{d}^{j+1}) = \mathbf{c}^j \oplus (d_{j+1} \cdot \boldsymbol{\pi}(d_{j+1})). \tag{3.1}$$

Dans (3.1), $\mathbf{d}^j = [d_1 \dots d_j, 0 \dots 0]$, $\boldsymbol{\pi}(d_j)$ correspond à la j-ième ligne de $\mathbf{\Pi}$, c'est-à-dire, au vecteur de parité associé à d_j, et $\oplus$ représente l'opérateur XOR (*ou exclusif*). Durant la phase d'initialisation, $\mathbf{c}^0$ est fixé à $\mathbf{0}$. Après $\ell(\mathbf{d})$ itérations, le vecteur $\mathbf{c}^{\ell(\mathbf{d})}$ contient la valeur du CRC associée à $\mathbf{d}$ ($\mathbf{c}^{\ell(\mathbf{d})} = \mathbf{c}$).

Le vecteur $\mathbf{t}$ est ensuite modulé en BPSK et transmis sur un canal de transmission AWGN (*Additive White Gaussian Noise*) de moyenne nulle et de variance σ^2. Pour faciliter les développements théoriques, nous négligeons le codage canal dans cette partie. Au niveau du récepteur, le signal reçu est défini par $\mathbf{y}_t = [\mathbf{y}_h, \mathbf{y}_x, \mathbf{y}_c]$, dans lequel $\mathbf{y}_h$, $\mathbf{y}_x$ et $\mathbf{y}_c$ correspondent aux observations respectives de $\mathbf{h}$, $\mathbf{x}$ et $\mathbf{c}$. Plus précisément, $\mathbf{y}_t$ contient les observations de $\mathbf{t}$ et représente un segment du paquet APL illustré sur la figure 3.5. Un aperçu du schéma de transmission est exposé sur la figure 3.6.

Dans ce qui suit, nous nous concentrons sur les traitements liés au décodage robuste de la vidéo. Nous cherchons à estimer $\mathbf{x}$ à partir de $\mathbf{y}_t$, en prenant en compte les propriétés du CRC et les redondances résiduelles contenues dans $\mathbf{x}$. De plus, nous faisons l'hypothèse que l'en-tête des paquets MAC est correctement reçu par le récepteur. L'estimateur optimal $\widehat{\mathbf{x}}$ au

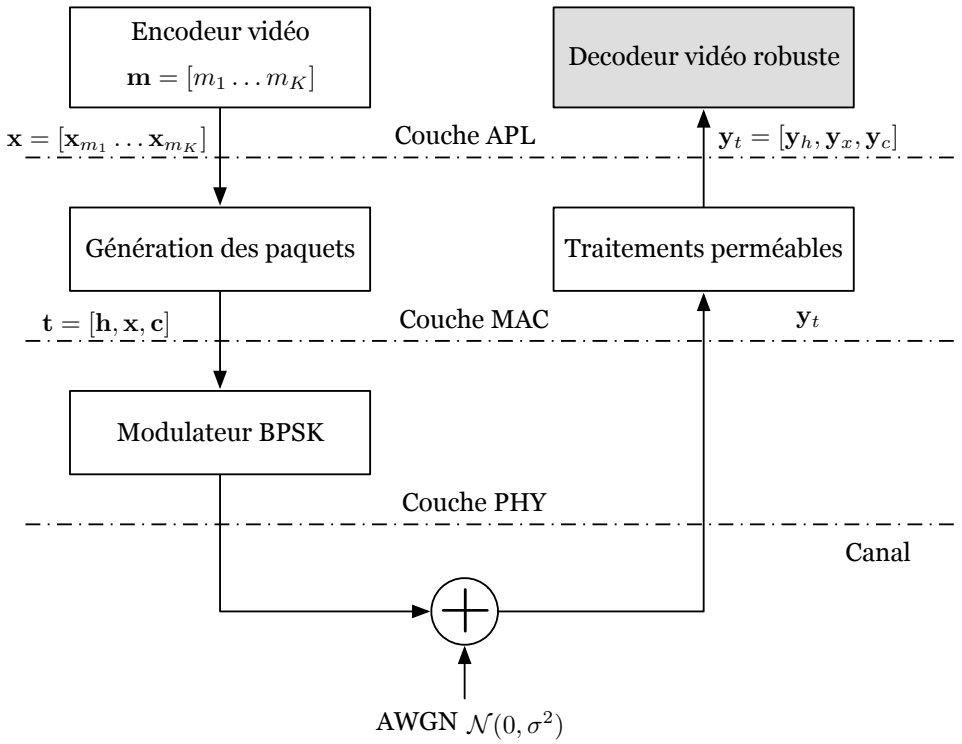


FIG. 3.6 – Illustration du schéma de transmission

sens MAP (*Maximum A Posteriori*) est donc donné par

$$\widehat{\mathbf{x}} = \arg\max_{\mathbf{x}\in\Omega_x} P(\mathbf{x}|\mathbf{h}, \mathbf{y}_x, \mathbf{y}_c), \tag{3.2}$$

où Ω_x contient l'ensemble des combinaisons de $\mathbf{x}$ respectant la syntaxe et la sémantique du codeur H.264/AVC. Dans (3.2), $P(\mathbf{x}|\mathbf{h}, \mathbf{y}_x, \mathbf{y}_c)$ représente la probabilité *a posteriori* de $\mathbf{x}$ connaissant $\mathbf{h}$, $\mathbf{y}_x$ et $\mathbf{y}_c$.

Néanmoins, les nombreuses combinaisons contenues dans Ω_x sont mal structurées. Pour des raisons de complexité, les $\ell(\mathbf{x})$ bits de $\mathbf{x}$ ne peuvent pas être décodés directement. C'est pourquoi, nous proposons un traitement de moindre complexité s'appuyant sur un décodage séquentiel de $\mathbf{x}$ [67]. Cet algorithme consiste à décoder $\mathbf{x}$ par portions successives. A chaque étape du décodage séquentiel, $\mathbf{x}$ est composé de trois nouvelles parties. A l'étape n, nous avons $\mathbf{x} = [\mathbf{u}_n, \mathbf{s}_n, \mathbf{r}_n]$ où :

- Le vecteur $\mathbf{u}_n$ est composé de $\ell(\mathbf{u}_n)$ bits pour lesquels un ensemble réduit de combinaisons Ω_u^n a été sauvegardé précédemment par le décodeur. Ces combinaisons correspondaient aux séquences les plus probables à l'étape $n-1$. Le cardinal de Ω_u^n est noté M ($M = |\Omega_u^n|$) et définit le nombre de combinaisons dans Ω_u^n.
- Le vecteur $\mathbf{s}_n$ est composé de $\ell(\mathbf{s}_n)$ bits pour lesquels, en négligeant la syntaxe et la sémantique du codeur vidéo, $2^{\ell(\mathbf{s}_n)}$ combinaisons sont possibles. Dans la suite, nous appelons Ω_s^n l'ensemble contenant la totalité de ces séquences.

– Le vecteur $\mathbf{r}_n$ contient les $\ell(\mathbf{r}_n)$ derniers bits de $\mathbf{x}$. Ces bits n'ont pas encore été traités par le décodeur, mais influencent le CRC.

Les observations associées à ces trois vecteurs sont contenues dans $\mathbf{y}_u^n$, $\mathbf{y}_s^n$ et $\mathbf{y}_r^n$. La figure 3.7 illustre le schéma de segmentation du paquet à l'étape n.

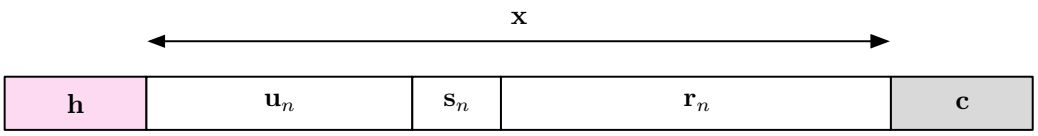


FIG. 3.7 – Segmentation d'un paquet lors du décodage séquentiel

A présent, considérons la concaténation de chacune des M séquences de Ω_u^n avec chacune des $2^{\ell(\mathbf{s}_n)}$ séquences de Ω_s^n. Nous aboutissons à $M \cdot 2^{\ell(\mathbf{s}_n)}$ séquences binaires pour le vecteur $[\mathbf{u}_n, \mathbf{s}_n]$. Cependant, parmi ces $M \cdot 2^{\ell(\mathbf{s}_n)}$ séquences, seule une partie respecte les propriétés sémantiques et syntaxiques imposées par le codeur. Toutes les séquences valides sont stockées dans l'ensemble $\Omega_{[u,s]}^n$. Parmi ces séquences valides, seules les M séquences les plus probables sont conservées par le décodeur séquentiel et définissent l'ensemble Ω_u^{n+1} qui sera utilisé à la prochaine étape. L'algorithme séquentiel répète ce procédé à chaque étape et se termine après avoir balayé tous les bits de $\mathbf{x}$. A chaque étape, les séquences les moins probables sont délaissées. Cet élagage dépend du paramètre M qui fixe le compromis entre efficacité et complexité.

Nous allons maintenant déterminer l'expression de la métrique liée au décodage séquentiel. A la n-ième étape, nous devons calculer la probabilité *a posteriori* de $[\mathbf{u}_n, \mathbf{s}_n]$ connaissant $\mathbf{y}_t$ et $\mathbf{h}$, c'est-à-dire $P(\mathbf{u}_n, \mathbf{s}_n | \mathbf{y}_u^n, \mathbf{y}_s^n, \mathbf{y}_r^n, \mathbf{y}_c, \mathbf{h})$, pour chaque $\mathbf{u}_n \in \Omega_u^n$ et chaque $\mathbf{s}_n \in \Omega_s^n$.

D'après Bayes, nous pouvons montrer que cette probabilité est donnée par

$$P(\mathbf{u}_n, \mathbf{s}_n | \mathbf{y}_u^n, \mathbf{y}_s^n, \mathbf{y}_r^n, \mathbf{y}_c, \mathbf{h}) = \frac{P(\mathbf{u}_n, \mathbf{s}_n, \mathbf{y}_u^n, \mathbf{y}_s^n, \mathbf{y}_r^n, \mathbf{y}_c | \mathbf{h})}{P(\mathbf{y}_u^n, \mathbf{y}_s^n, \mathbf{y}_r^n, \mathbf{y}_c)} \propto P(\mathbf{u}_n, \mathbf{s}_n, \mathbf{y}_u^n, \mathbf{y}_s^n, \mathbf{y}_r^n, \mathbf{y}_c | \mathbf{h}). \quad (3.3)$$

Dans (3.3), $P(\mathbf{u}_n, \mathbf{s}_n, \mathbf{y}_u^n, \mathbf{y}_s^n, \mathbf{y}_r^n, \mathbf{y}_c | \mathbf{h})$ peut être écrite comme suit

$$P(\mathbf{u}_n, \mathbf{s}_n, \mathbf{y}_u^n, \mathbf{y}_s^n, \mathbf{y}_r^n, \mathbf{y}_c | \mathbf{h}) = \sum_{\mathbf{r}_n} P(\mathbf{u}_n, \mathbf{s}_n, \mathbf{r}_n, \mathbf{y}_u^n, \mathbf{y}_s^n, \mathbf{y}_r^n, \mathbf{y}_c | \mathbf{h}). \quad (3.4)$$

D'après (3.4), nous constatons que $P(\mathbf{u}_n, \mathbf{s}_n, \mathbf{y}_u^n, \mathbf{y}_s^n, \mathbf{y}_r^n, \mathbf{y}_c | \mathbf{h})$ peut être obtenue en marginalisant $P(\mathbf{u}_n, \mathbf{s}_n, \mathbf{r}_n, \mathbf{y}_u^n, \mathbf{y}_s^n, \mathbf{y}_r^n, \mathbf{y}_c | \mathbf{h})$ sur les $2^{\ell(\mathbf{r}_n)}$ combinaisons de $\mathbf{r}_n$. Par ailleurs, nous pouvons développer $P(\mathbf{u}_n, \mathbf{s}_n, \mathbf{r}_n, \mathbf{y}_u^n, \mathbf{y}_s^n, \mathbf{y}_r^n, \mathbf{y}_c | \mathbf{h})$ tel que

$$P(\mathbf{u}_n, \mathbf{s}_n, \mathbf{r}_n, \mathbf{y}_u^n, \mathbf{y}_s^n, \mathbf{y}_r^n, \mathbf{y}_c | \mathbf{h}) = P(\mathbf{u}_n, \mathbf{s}_n | \mathbf{h}) P(\mathbf{y}_u^n, \mathbf{y}_s^n | \mathbf{h}, \mathbf{u}_n, \mathbf{s}_n) P(\mathbf{r}_n, \mathbf{y}_r^n, \mathbf{y}_c | \mathbf{h}, \mathbf{u}_n, \mathbf{s}_n, \mathbf{y}_u^n, \mathbf{y}_s^n). \quad (3.5)$$

Suite aux conditions d'indépendance entre les variables, (3.5) peut être réécrite comme suit

$$P(\mathbf{u}_n, \mathbf{s}_n, \mathbf{r}_n, \mathbf{y}_u^n, \mathbf{y}_s^n, \mathbf{y}_r^n, \mathbf{y}_c | \mathbf{h}) = P(\mathbf{u}_n, \mathbf{s}_n) P(\mathbf{y}_u^n | \mathbf{u}_n) P(\mathbf{y}_s^n | \mathbf{s}_n) P(\mathbf{r}_n, \mathbf{y}_r^n, \mathbf{y}_c | \mathbf{h}, \mathbf{u}_n, \mathbf{s}_n). \quad (3.6)$$

Finalement, en combinant (3.3), (3.4) et (3.6), nous obtenons

$$\begin{aligned} P(\mathbf{u}_n, \mathbf{s}_n | \mathbf{y}_u^n, \mathbf{y}_s^n, \mathbf{y}_r^n, \mathbf{y}_c, \mathbf{h}) &\propto P(\mathbf{u}_n, \mathbf{s}_n) P(\mathbf{y}_u^n | \mathbf{u}_n) P(\mathbf{y}_s^n | \mathbf{s}_n) \sum_{\mathbf{r}_n} P(\mathbf{r}_n, \mathbf{y}_r^n, \mathbf{y}_c | \mathbf{h}, \mathbf{u}_n, \mathbf{s}_n) \\ &= P(\mathbf{u}_n, \mathbf{s}_n) P(\mathbf{y}_u^n | \mathbf{u}_n) P(\mathbf{y}_s^n | \mathbf{s}_n) \Psi(\mathbf{h}, \mathbf{u}_n, \mathbf{s}_n, \mathbf{y}_r^n, \mathbf{y}_c), \end{aligned} \tag{3.7}$$

avec

$$\Psi(\mathbf{h}, \mathbf{u}_n, \mathbf{s}_n, \mathbf{y}_r^n, \mathbf{y}_c) = \sum_{\mathbf{r}_n} P(\mathbf{r}_n, \mathbf{y}_r^n, \mathbf{y}_c | \mathbf{h}, \mathbf{u}_n, \mathbf{s}_n). \tag{3.8}$$

Dans (3.6), $P(\mathbf{u}_n, \mathbf{s}_n)$ représente la probabilité *a priori* de la séquence $[\mathbf{u}_n, \mathbf{s}_n]$. Cette probabilité est nulle quand le vecteur binaire $[\mathbf{u}_n, \mathbf{s}_n]$ n'a pas pu être généré par le codeur H.264/AVC. En d'autres termes, $P(\mathbf{u}_n, \mathbf{s}_n) = 0$ si $[\mathbf{u}_n, \mathbf{s}_n]$ ne respecte pas les conditions sémantiques et syntaxiques fixées par le codeur source. Dans le cas contraire, c'est-à-dire quand $[\mathbf{u}_n, \mathbf{s}_n]$ correspond à une séquence valide, $P(\mathbf{u}_n, \mathbf{s}_n)$ est non-nulle. Dans notre étude, nous considérons que toutes les séquences valides (contenues dans $\Omega_{[u,s]}^n$) sont équiprobables et $P(\mathbf{u}_n, \mathbf{s}_n) = 1/|\Omega_{[u,s]}^n|$ pour toutes les séquences $[\mathbf{u}_n, \mathbf{s}_n] \in \Omega_{[u,s]}^n$. Par conséquent, la métrique de décodage $\mathcal{M}$ associée à une séquence *valide* est donnée par

$$\mathcal{M}([\mathbf{u}_n, \mathbf{s}_n] \in \Omega_{[u,s]}^n | \mathbf{h}, \mathbf{y}_t) = P(\mathbf{y}_u^n | \mathbf{u}_n) P(\mathbf{y}_s^n | \mathbf{s}_n) \Psi(\mathbf{h}, \mathbf{u}_n, \mathbf{s}_n, \mathbf{y}_r^n, \mathbf{y}_c). \tag{3.9}$$

Dans (3.9), $P(\mathbf{y}_u^n | \mathbf{u}_n)$ et $P(\mathbf{y}_s^n | \mathbf{s}_n)$ représentent les vraisemblances respectives de $\mathbf{u}_n$ et $\mathbf{s}_n$. $\Psi(\mathbf{h}, \mathbf{u}_n, \mathbf{s}_n, \mathbf{y}_r^n, \mathbf{y}_c)$ correspond à une somme dont la complexité est exponentielle en $\ell(\mathbf{r}_n)$.

Nous allons maintenant décrire le fonctionnement complet du décodage séquentiel. Durant la phase d'initialisation ($n = 1$), $\Omega_u^1 = \emptyset$. Puis, à chaque étape $n > 1$, l'algorithme explore les nouvelles branches possibles sur une profondeur de $\ell(\mathbf{s}_n)$ bits. Il conserve seulement les M séquences étendues $[\mathbf{u}_n, \mathbf{s}_n]$ les plus probables. Ces M séquences sont temporairement stockées dans une mémoire (correspondant à Ω_u^{n+1}) avant d'être étendues à nouveau à la prochaine étape. Le synoptique de décodage est représenté sur la figure 3.8. Précisons juste que la métrique $\mathcal{M}([\mathbf{u}_n, \mathbf{s}_n] \in \Omega_{[u,s]}^n | \mathbf{h}, \mathbf{y}_t)$ est déterminée à partir de (3.9).

3.4 Principe optimisé de décodage séquentiel

Dans la partie précédente, nous avons présenté une méthode générale de décodage séquentiel pouvant s'appliquer à tous les flux comprimés. Cependant, cet algorithme peut facilement diverger et conduire à une solution incorrecte. En effet, si la séquence correcte n'est pas conservée dans la mémoire à une certaine étape de décodage, la solution finale sera toujours erronée. Lorsque les paquets transmis ont une taille importante, cette situation peut apparaître très fréquemment et la solution finale peut contenir beaucoup d'erreurs binaires.

En pratique, $\mathbf{x}$ est généralement composé de groupes de mots de code, qui sont encodés indépendamment les uns des autres. Pour améliorer les performances du décodage, il semble intéressant de transmettre au récepteur les longueurs relatives des différents groupes et d'appliquer le décodage sur chaque groupe séparément. Dans la suite, les vecteurs $\mathbf{a}_1 \dots \mathbf{a}_E$ caractérisent les groupes de mots de code inclus dans $\mathbf{x}$ et nous pouvons écrire $\mathbf{x} = [\mathbf{a}_1 \dots \mathbf{a}_E]$. Par ailleurs, $\ell(\mathbf{a}_e)$ désigne la longueur en bits du groupe $\mathbf{a}_e$, $e = 1 \dots E$.

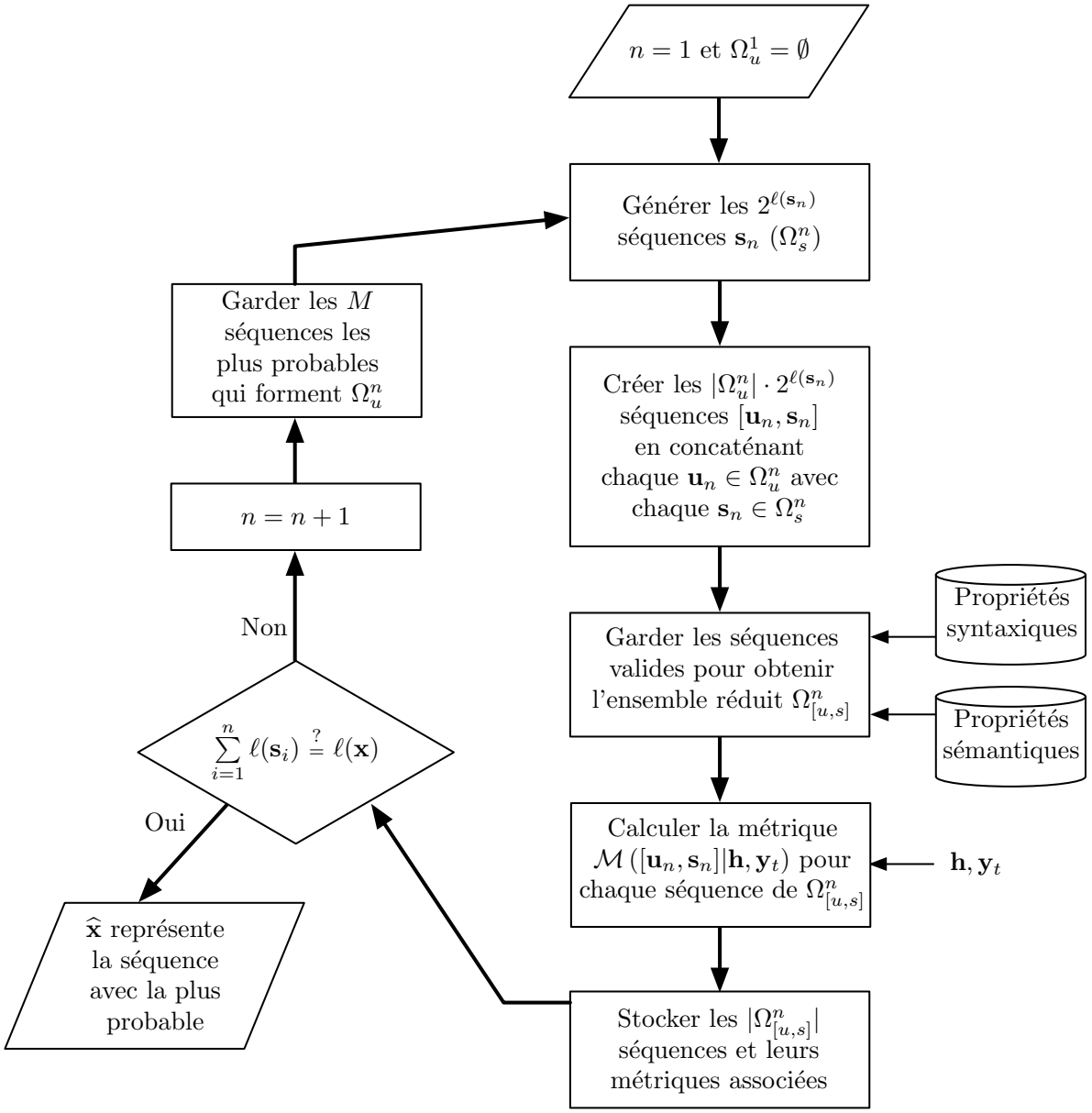


FIG. 3.8 – Description en blocs du décodage séquentiel de $\mathbf{x}$

Les marqueurs de synchronisation $\ell(\mathbf{a}_1) \dots \ell(\mathbf{a}_E)$ sont transmis sur un canal de transmission séparé (*side information*). Lors du décodage robuste, ces marqueurs sont utilisés pour localiser chaque groupe dans le paquet. A titre d'illustration, ces marqueurs pourraient être transmis dans un paquet NAL spécifique, garantissant ainsi la compatibilité avec la norme H.264/AVC.

A présent, nous allons nous intéresser au décodage du groupe $\mathbf{a}_e$, $e = 1 \dots E$. Les groupes $\mathbf{a}_1 \dots \mathbf{a}_{e-1}$ ont été décodés précédemment et peuvent contenir des erreurs. Ces erreurs potentielles peuvent impacter le décodage de $\mathbf{a}_e$ lors de l'exploitation du CRC. Afin d'éviter la

propagation des erreurs entre les groupes, le décodage de $\mathbf{a}_e$ ne tient pas compte des résultats de décodage de $\mathbf{a}_1 \dots \mathbf{a}_{e-1}$. L'estimateur optimal $\widehat{\mathbf{a}}_e$ au sens MAP est donc donné par

$$\widehat{\mathbf{a}}_e = \arg \max_{\mathbf{a}_e \in \Omega_a^e} P(\mathbf{a}_e | \mathbf{h}, \mathbf{y}_x, \mathbf{y}_c), \tag{3.10}$$

où Ω_a^e contient l'ensemble des combinaisons de $\mathbf{a}_e$ respectant la syntaxe et la sémantique du codeur source. Dans (3.10), $P(\mathbf{a}_e|\mathbf{h}, \mathbf{y}_x, \mathbf{y}_c)$ représente la probabilité *a posteriori* de $\mathbf{a}_e$ connaissant $\mathbf{h}$, $\mathbf{y}_x$ et $\mathbf{y}_c$.

Cependant, les combinaisons contenues dans Ω_a^e sont mal structurées et les $\ell(\mathbf{a}_e)$ bits ne peuvent pas être directement décodés pour des raisons de complexité. De manière similaire à la partie 3.3, le décodage s'effectue sur des portions de $\mathbf{a}_e$. Le décodage séquentiel se limite donc à un groupe de mots de code et n'accorde aucune confiance aux solutions obtenues lors du décodage des groupes précédents. Considérons maintenant le décodage séquentiel du groupe e à la n-ième étape, nous avons $\mathbf{x} = [\mathbf{b}_e, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{r}_{e,n}]$, dans lequel :

- Le vecteur $\mathbf{b}_e$ correspond aux $\ell(\mathbf{b}_e) = \sum_{i=1}^{e-1} \ell(\mathbf{a}_i)$ bits associés aux $e-1$ premiers groupes de $\mathbf{x}$. Lors du décodage de $\mathbf{a}_e$, $\mathbf{b}_e$ est considéré comme un vecteur de bits aléatoires : il ne contient pas les solutions précédemment décodées de $\mathbf{a}_1 \dots \mathbf{a}_{e-1}$.
- Le vecteur $\mathbf{u}_{e,n}$ correspond aux $\ell(\mathbf{u}_{e,n})$ premiers bits de $\mathbf{a}_e$ pour lesquels un ensemble réduit de combinaisons $\Omega_u^{e,n}$ a été précédemment sauvegardé par le décodeur. Ces combinaisons représentaient les portions les plus probables de $\mathbf{a}_e$ à l'étape $n-1$. Dans la suite, M définit le cardinal de $\Omega_u^{e,n}$ ($M = |\Omega_u^{e,n}|$).
- Le vecteur $\mathbf{s}_{e,n}$ est composé de $\ell(\mathbf{s}_{e,n})$ bits pour lesquels, en négligeant les propriétés structurelles du codeur vidéo, $2^{\ell(\mathbf{s}_{e,n})}$ combinaisons sont possibles. Dans la suite, nous nommons $\Omega_s^{e,n}$ l'ensemble contenant la totalité de ces combinaisons.
- Le vecteur $\mathbf{r}_{e,n}$ contient les $\ell(\mathbf{r}_{e,n})$ derniers bits de $\mathbf{x}$. Ces bits n'ont pas encore été traités par le décodeur, mais interviennent dans le calcul du CRC.

Les observations associées à ces quatre vecteurs sont contenues dans les vecteurs $\mathbf{y}_b^e$, $\mathbf{y}_u^{e,n}$, $\mathbf{y}_s^{e,n}$ et $\mathbf{y}_r^{e,n}$. Le nouveau schéma de segmentation du paquet est exposé sur la figure 3.9.

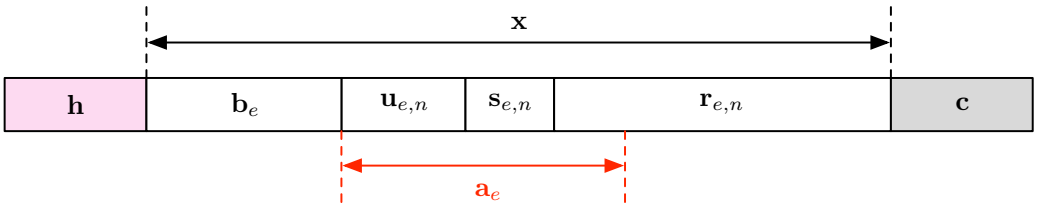


FIG. 3.9 – Nouveau schéma de segmentation du paquet

A présent, considérons la concaténation de chacune des M séquences de $\Omega_u^{e,n}$ avec chacune des $2^{\ell(\mathbf{s}_{e,n})}$ séquences de $\Omega_s^{e,n}$. Nous obtenons $M \cdot 2^{\ell(\mathbf{s}_{e,n})}$ séquences binaires pour le vecteur $[\mathbf{u}_{e,n}, \mathbf{s}_{e,n}]$. Cependant, seul un ensemble réduit $\Omega_{[u,s]}^{e,n}$ de ces séquences est valide, c'est-à-dire, respecte les propriétés sémantiques et syntaxiques du codeur. A la fin de chaque étape, le

décodeur conserve seulement les M séquences les plus probables de $\Omega^{e,n}_{[u,s]}$, les stocke dans l'ensemble $\Omega^{e,n+1}_{u}$ et commence un nouveau cycle. La figure 3.10 illustre l'évolution des portions $\mathbf{b}_e$, $\mathbf{u}_{e,n}$, $\mathbf{s}_{e,n}$ et $\mathbf{r}_{e,n}$ en fonction des étapes de décodage.

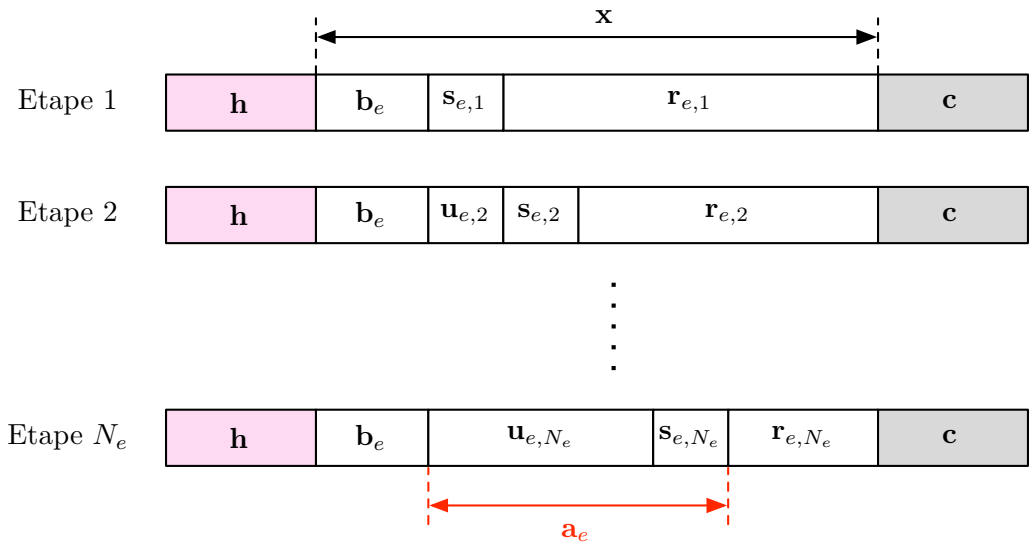


FIG. 3.10 – Evolution des partitions en fonction des étapes de décodage séquentiel

Dans la suite, nous considérons que N_e représente le nombre d'étapes nécessaires pour atteindre la fin du groupe e. Les profondeurs de décodage $\ell(\mathbf{s}_{e,i})$, pour $i = 1 \dots N_e$, peuvent donc être ajustées tel que

$$\sum_{i=1}^{N_e} \ell(\mathbf{s}_{e,i}) = \ell(\mathbf{a}_e), \tag{3.11}$$

pour tous $e \in \{1 \dots E\}$. En pratique, les $N_e - 1$ premières profondeurs sont fixées à une valeur paramétrique constante et la dernière $(\ell(\mathbf{s}_{e,N_e}))$ est choisie de manière à satisfaire (3.11).

Nous allons maintenant déterminer l'expression de la métrique associée à cette nouvelle méthode de décodage. A la n-ième étape, nous devons calculer la probabilité *a posteriori* de $[\mathbf{u}_{e,n}, \mathbf{s}_{e,n}]$ connaissant $\mathbf{y}_t$ et $\mathbf{h}$, c'est-à-dire, $P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n} | \mathbf{y}^e_b, \mathbf{y}^{e,n}_u, \mathbf{y}^{e,n}_s, \mathbf{y}^{e,n}_r, \mathbf{y}_c, \mathbf{h})$, pour chaque $\mathbf{u}_{e,n} \in \Omega^{e,n}_u$ et chaque $\mathbf{s}_{e,n} \in \Omega^{e,n}_s$.

D'après Bayes, cette probabilité est donnée par

$$P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n} | \mathbf{y}^e_b, \mathbf{y}^{e,n}_u, \mathbf{y}^{e,n}_s, \mathbf{y}^{e,n}_r, \mathbf{y}_c, \mathbf{h}) \propto P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{y}^e_b, \mathbf{y}^{e,n}_u, \mathbf{y}^{e,n}_s, \mathbf{y}^{e,n}_r, \mathbf{y}_c | \mathbf{h}). \tag{3.12}$$

Dans (3.12), $P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{y}^e_b, \mathbf{y}^{e,n}_u, \mathbf{y}^{e,n}_s, \mathbf{y}^{e,n}_r, \mathbf{y}_c | \mathbf{h})$ peut être écrite comme suit

$$P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{y}^e_b, \mathbf{y}^{e,n}_u, \mathbf{y}^{e,n}_s, \mathbf{y}^{e,n}_r, \mathbf{y}_c | \mathbf{h}) = \sum_{\mathbf{b}_e} \sum_{\mathbf{r}_{e,n}} P(\mathbf{b}_e, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{r}_{e,n}, \mathbf{y}^e_b, \mathbf{y}^{e,n}_u, \mathbf{y}^{e,n}_s, \mathbf{y}^{e,n}_r, \mathbf{y}_c | \mathbf{h}). \tag{3.13}$$

D'après (3.13), nous pouvons remarquer que $P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_u^{e,n}, \mathbf{y}_s^{e,n}, \mathbf{y}_r^{e,n}, \mathbf{y}_c|\mathbf{h})$ peut être déterminée en marginalisant $P(\mathbf{b}_e, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{r}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_u^{e,n}, \mathbf{y}_s^{e,n}, \mathbf{y}_r^{e,n}, \mathbf{y}_c|\mathbf{h})$ sur les $2^{\ell(\mathbf{b}_e)}$ combinaisons de $\mathbf{b}_e$ et sur les $2^{\ell(\mathbf{r}_{e,n})}$ combinaisons de $\mathbf{r}_{e,n}$. Par ailleurs, nous pouvons développer $P(\mathbf{b}_e, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{r}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_u^{e,n}, \mathbf{y}_s^{e,n}, \mathbf{y}_r^{e,n}, \mathbf{y}_c|\mathbf{h})$ en utilisant des conditions d'indépendance tel que

$$\begin{aligned} &P(\mathbf{b}_e, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{r}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_u^{e,n}, \mathbf{y}_s^{e,n}, \mathbf{y}_r^{e,n}, \mathbf{y}_c|\mathbf{h}) \\ &\quad = P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n}) P(\mathbf{y}_u^{e,n}|\mathbf{u}_{e,n}) P(\mathbf{y}_s^{e,n}|\mathbf{s}_{e,n}) P(\mathbf{b}_e, \mathbf{r}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_r^{e,n}, \mathbf{y}_c|\mathbf{h}, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}). \end{aligned} \tag{3.14}$$

Finalement, en combinant (3.12), (3.13) et (3.14), nous aboutissons à

$$\begin{aligned} &P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n}|\mathbf{y}_b^e, \mathbf{y}_u^{e,n}, \mathbf{y}_s^{e,n}, \mathbf{y}_r^{e,n}, \mathbf{y}_c, \mathbf{h}) \\ &\quad \propto P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n}) P(\mathbf{y}_u^{e,n}|\mathbf{u}_{e,n}) P(\mathbf{y}_s^{e,n}|\mathbf{s}_{e,n}) \Phi(\mathbf{h}, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_r^{e,n}, \mathbf{y}_c), \end{aligned} \tag{3.15}$$

avec

$$\Phi(\mathbf{h}, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_r^{e,n}, \mathbf{y}_c) = \sum_{\mathbf{b}_e, \mathbf{r}_{e,n}} P(\mathbf{b}_e, \mathbf{r}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_r^{e,n}, \mathbf{y}_c|\mathbf{h}, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}). \tag{3.16}$$

Dans (3.15), $P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n})$ représente la probabilité *a priori* de la séquence $[\mathbf{u}_{e,n}, \mathbf{s}_{e,n}]$. Cette probabilité est nulle quand le vecteur binaire $[\mathbf{u}_{e,n}, \mathbf{s}_{e,n}]$ n'a pas pu être généré par le codeur source. Dans le cas contraire, nous considérons dans cette étude que toutes les séquences valides sont équiprobables. $P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n})$ peut donc être synthétisée comme suit

$$P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n}) = \begin{cases} 0 & \text{si } [\mathbf{u}_{e,n}, \mathbf{s}_{e,n}] \notin \Omega_{[u,s]}^{e,n} \\ \frac{1}{|\Omega_{[u,s]}^{e,n}|} & \text{si } [\mathbf{u}_{e,n}, \mathbf{s}_{e,n}] \in \Omega_{[u,s]}^{e,n} \end{cases}.$$

Par conséquent, la métrique de décodage associée à une séquence *valide* pour le groupe e est donnée par

$$\mathcal{M}_e([\mathbf{u}_{e,n}, \mathbf{s}_{e,n}] \in \Omega_{[u,s]}^{e,n}|\mathbf{h}, \mathbf{y}_t) = P(\mathbf{y}_u^{e,n}|\mathbf{u}_{e,n}) P(\mathbf{y}_s^{e,n}|\mathbf{s}_{e,n}) \Phi(\mathbf{h}, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_r^{e,n}, \mathbf{y}_c). \tag{3.17}$$

Dans (3.17), $P(\mathbf{y}_u^{e,n}|\mathbf{u}_{e,n})$ et $P(\mathbf{y}_s^{e,n}|\mathbf{s}_{e,n})$ représentent les vraisemblances respectives de $\mathbf{u}_{e,n}$ et $\mathbf{s}_{e,n}$. $\Phi(\mathbf{h}, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_r^{e,n}, \mathbf{y}_c)$ correspond à une somme dont la complexité est $\mathcal{O}(2^{\ell(\mathbf{b}_e)+\ell(\mathbf{r}_{e,n})})$. La réduction de la complexité de cette somme est discutée dans la partie 3.5.

Le synoptique associé à la nouvelle méthode de décodage est exposé sur la figure 3.11. Nous pouvons remarquer de fortes corrélations entre les figures 3.8 et 3.11. Le fonctionnement de l'algorithme séquentiel reste identique. La différence principale est associée à la portée du décodage : dans la nouvelle méthode, le décodage n'est plus directement réalisé sur tous les bits de $\mathbf{x}$ mais sur un groupe $\mathbf{a}_e$ ($e = 1 \dots E$).

Dans la partie 3.6, cet algorithme est appliqué au décodage des séquences CAVLC générées par le codeur H.264/AVC.

3.5 Réduction de la complexité algorithmique

Pour faciliter la lecture du document, les indices e et n ont été volontairement retirés des équations dans la suite de cette partie. Dans le même registre, $\Phi(\mathbf{h}, \mathbf{u}, \mathbf{s}, \mathbf{y}_b, \mathbf{y}_r, \mathbf{y}_c)$ et $\mathcal{M}([\mathbf{u}, \mathbf{s}] \in \Omega_{[u,s]}|\mathbf{h}, \mathbf{y}_t)$ ont été remplacés par Φ et $\mathcal{M}([\mathbf{u}, \mathbf{s}])$ respectivement.

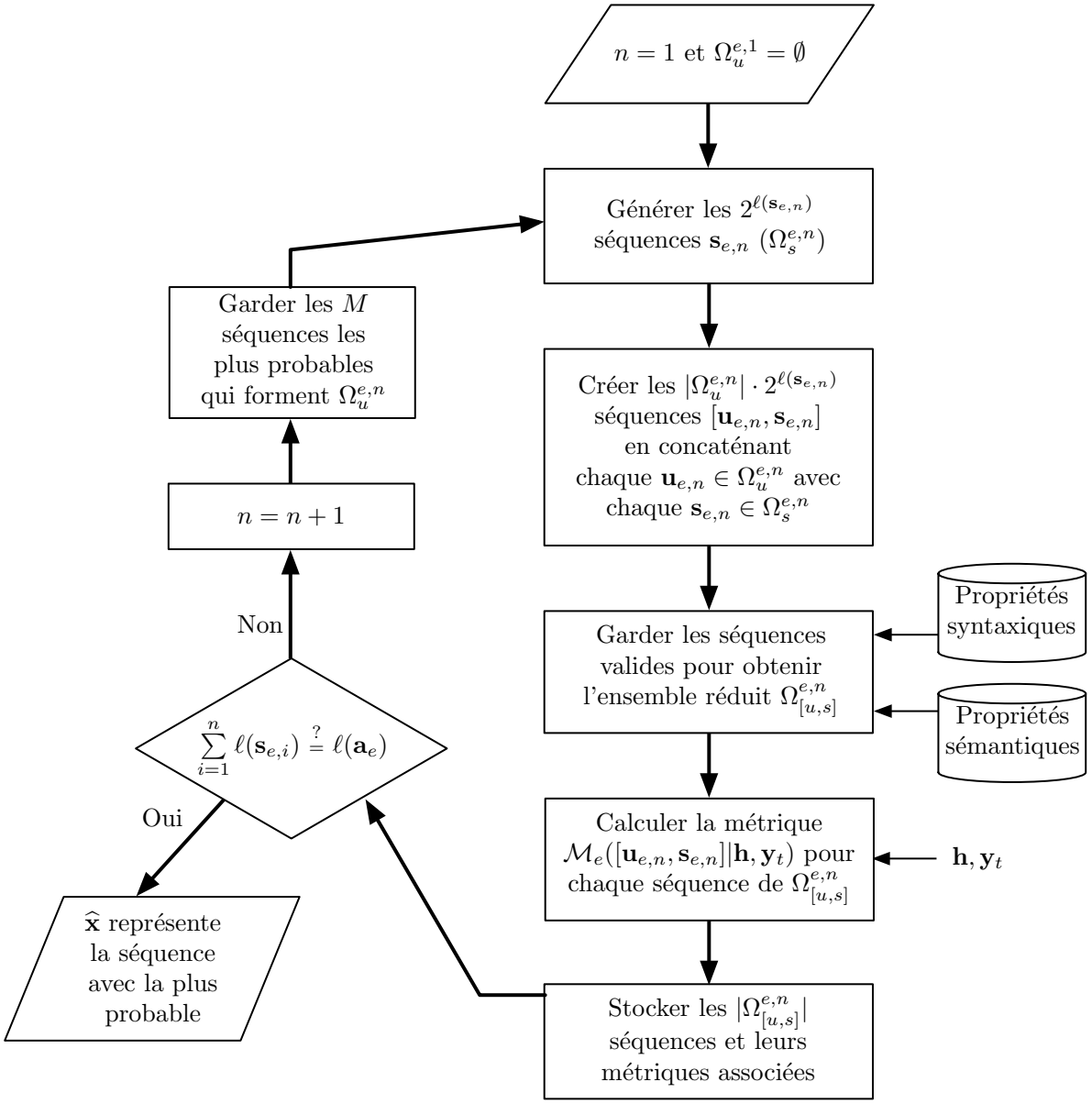


FIG. 3.11 – Schéma blocs de décodage du groupe $\mathbf{a}_e$

Dans (3.17), seul le terme Φ est complexe à déterminer. En considérant que les bits de $\mathbf{b}$ et $\mathbf{r}$ sont i.i.d. et ne dépendent pas de $\mathbf{h}$, $\mathbf{u}$ et $\mathbf{s}$, (3.16) devient

$$\Phi = \sum_{\mathbf{b}} \sum_{\mathbf{r}} P(\mathbf{b}) P(\mathbf{y}_b|\mathbf{b}) P(\mathbf{r}) P(\mathbf{y}_r|\mathbf{r}) P(\mathbf{y}_c|\mathcal{F}([\mathbf{h}, \mathbf{b}, \mathbf{u}, \mathbf{s}, \mathbf{r}])). \tag{3.18}$$

Dans (3.18), $P(\mathbf{b})$ et $P(\mathbf{r})$ représentent les probabilités *a priori* de $\mathbf{b}$ et $\mathbf{r}$. En suivant les hypothèses d'indépendances énoncées ci-dessus, $P(\mathbf{b}) = 2^{-\ell(\mathbf{b})}$ et $P(\mathbf{r}) = 2^{-\ell(\mathbf{r})}$. $P(\mathbf{y}_b|\mathbf{b})$, $P(\mathbf{y}_r|\mathbf{r})$ et $P(\mathbf{y}_c|\mathcal{F}([\mathbf{h}, \mathbf{b}, \mathbf{u}, \mathbf{s}, \mathbf{r}]))$ définissent les vraisemblances respectives de $\mathbf{b}$, $\mathbf{r}$ et $\mathbf{c} = \mathcal{F}([\mathbf{h}, \mathbf{b}, \mathbf{u}, \mathbf{s}, \mathbf{r}])$.

Le calcul de (3.18) consiste à sommer le produit des vraisemblances associées à $\mathbf{b}$, $\mathbf{r}$ et leur CRC correspondant, sur les $2^{\ell(\mathbf{b})+\ell(\mathbf{r})}$ combinaisons de $\mathbf{b}$ et $\mathbf{r}$. Dans cette partie, deux méthodes de complexité réduite sont proposées pour déterminer (3.17) en optimisant le calcul de Φ. La première méthode fournit un calcul exact de la métrique, tandis que la deuxième délivre une solution approchée pour une plus faible complexité algorithmique.

3.5.1 Calcul exact de la métrique

D'après (3.1), le CRC peut être déterminé de manière récursive sur les bits de $\mathbf{d}$. Plus précisément, la valeur du CRC associée aux $j+1$ premiers bits de $\mathbf{d}$ dépend seulement de la valeur du CRC à l'instant j et du $j+1$-ième bit de $\mathbf{d}$. Chaque valeur de CRC à l'instant j peut conduire à deux différentes valeurs de CRC à l'instant $j+1$. Par conséquent, l'évolution des valeurs du CRC en fonction des bits de $\mathbf{d}$ peut être modélisée par un treillis. Dans ce treillis, les états correspondent aux $2^{\ell(\mathbf{c})}$ valeurs de CRC possibles et les transitions sont déterminées par les bits de $\mathbf{d}$. A chaque instant $j = 1 \dots \ell(\mathbf{d})$, nous pouvons étudier la contribution de d_j (j-ième bit de $\mathbf{d}$) sur le CRC global.

Dans notre cas, $\mathbf{d} = [\mathbf{h}, \mathbf{b}, \mathbf{u}, \mathbf{s}, \mathbf{r}]$. L'en-tête $\mathbf{h}$ est considéré connu et nous cherchons à trouver la meilleure combinaison de $[\mathbf{u}, \mathbf{s}] \in \Omega_{[u,s]}$ en tenant compte des redondances introduites par le CRC $\mathbf{c}$. Le treillis est donc appliqué aux portions $\mathbf{b}$, $\mathbf{r}$ et $\mathbf{c}$ pour des valeurs fixes de $\mathbf{h}$, $\mathbf{u}$ et $\mathbf{s}$. Ce treillis consiste à regrouper les combinaisons de $\mathbf{b}$ et $\mathbf{r}$ qui donnent les mêmes valeurs de CRC. En utilisant cette modélisation, (3.18) devient

$$\Phi = \sum_{\mathbf{c}} P(\mathbf{y}_c|\mathbf{c}) \sum_{\mathbf{b},\mathbf{r}|\mathcal{F}([\mathbf{h},\mathbf{b},\mathbf{u},\mathbf{s},\mathbf{r}])=\mathbf{c}} P(\mathbf{b})P(\mathbf{y}_b|\mathbf{b})P(\mathbf{r})P(\mathbf{y}_r|\mathbf{r}). \tag{3.19}$$

Dans la suite, nous considérons que l'état associé à une valeur possible de CRC $\mathbf{c}'$ est noté $S(\mathbf{c}')$, $\mathbf{c}'$ étant la représentation binaire de $S(\mathbf{c}') \in \{0 \dots 2^{\ell(\mathbf{c})} - 1\}$. Après quelques démonstrations, on peut démontrer que (3.19) peut être reformulé par

$$\begin{aligned} \Phi &= \sum_{\mathbf{c}'} \left[\sum_{\mathbf{b}|\mathcal{F}([\mathbf{h},\mathbf{b},\mathbf{0},\mathbf{0},\mathbf{0}])=\mathbf{c}'} P(\mathbf{b})P(\mathbf{y}_b|\mathbf{b}) \right] \left[\sum_{\mathbf{r}} P(\mathbf{r})P(\mathbf{y}_r|\mathbf{r})P(\mathbf{y}_c|\mathbf{c}' \oplus \mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{r}])) \right] \\ &= \sum_{\mathbf{c}'} \alpha(S(\mathbf{c}')) \cdot \beta(S(\mathbf{c}' \oplus \mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{0}]))), \end{aligned} \tag{3.20}$$

avec

$$\alpha(S(\mathbf{c}')) = \sum_{\mathbf{b}|\mathcal{F}([\mathbf{h},\mathbf{b},\mathbf{0},\mathbf{0},\mathbf{0}])=\mathbf{c}'} P(\mathbf{b})P(\mathbf{y}_b|\mathbf{b}), \tag{3.21}$$

$$\beta(S(\mathbf{c}'')) = \sum_{\mathbf{r}} P(\mathbf{r})P(\mathbf{y}_r|\mathbf{r})P(\mathbf{y}_c|\mathbf{c}'' \oplus \mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{r}])), \tag{3.22}$$

pour tout $\mathbf{c}', \mathbf{c}'' \in GF(2)^{\ell(\mathbf{c})}$. Dans (3.21), $\alpha(S(\mathbf{c}'))$ représente la somme des probabilités associées aux combinaisons de $\mathbf{b}$ qui atteignent l'état $S(\mathbf{c}')$ en commençant par l'état

$S(\mathcal{F}([\mathbf{h},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}]))$ dans le treillis sur $\mathbf{b}$. Dans (3.22), $\beta(S(\mathbf{c}''))$ représente la somme des probabilités associées à toutes les combinaisons de $[\mathbf{r},\mathbf{c}=\mathbf{c}''\oplus\mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{r}])]$ qui partent de l'état $S(\mathbf{c}'')$ dans le treillis sur $\mathbf{r}$.

Finalement, en intégrant (3.20) dans (3.17), on aboutit à

$$\begin{aligned}\mathcal{M}([\mathbf{u},\mathbf{s}]) &= \sum_{\mathbf{c}'}\alpha(S(\mathbf{c}'))\cdot P(\mathbf{y}_u|\mathbf{u})P(\mathbf{y}_s|\mathbf{s})\cdot\beta(S(\mathbf{c}'\oplus\mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{0}])))\\ &= P(\mathbf{y}_u|\mathbf{u})P(\mathbf{y}_s|\mathbf{s})\sum_{\mathbf{c}',\mathbf{c}''|\mathbf{c}''=\mathbf{c}'\oplus\mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{0}])}\alpha(S(\mathbf{c}'))\cdot\beta(S(\mathbf{c}'')).\end{aligned}\tag{3.23}$$

La détermination de $\mathcal{M}([\mathbf{u},\mathbf{s}])$ consiste donc à sommer les probabilités associées aux $2^{\ell(\mathbf{c})}$ chemins qui relient l'état $S(\mathbf{c}')$ du treillis sur $\mathbf{b}$ à l'état $S(\mathbf{c}'')$ du treillis sur $\mathbf{r}$, tel que $\mathbf{c}''=\mathbf{c}'\oplus\mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{0}])$.

A présent, nous allons montrer que $\alpha(S(\mathbf{c}'))$ dans (3.21) peut être déterminé de manière récursive sur les $\ell(\mathbf{b})$ bits de $\mathbf{b}$, pour tout $\mathbf{c}'\in GF(2)^{\ell(\mathbf{c})}$. Pour cela, nous considérons que $\mathbf{b}^j=[b_1\dots b_j,0\dots0]$ et $\mathbf{y}_b^j=[y_{b_1}\dots y_{b_j},0\dots0]$. De plus, nous définissons

$$\alpha_j(S(\mathbf{c}'))=\sum_{\mathbf{b}^j|\mathcal{F}([\mathbf{h},\mathbf{b}^j,\mathbf{0},\mathbf{0},\mathbf{0}])=\mathbf{c}'}P(\mathbf{b}^j)P(\mathbf{y}_b^j|\mathbf{b}^j),\tag{3.24}$$

comme la probabilité associée à l'état $S(\mathbf{c}')\in\{0\dots2^{\ell(\mathbf{c})}-1\}$ à l'instant $j\in\{0\dots\ell(\mathbf{b})\}$ dans le treillis sur $\mathbf{b}$.

En appliquant (3.24) à l'état $S(\mathbf{c}')$ à l'instant $j+1$, nous obtenons

$$\begin{aligned}\alpha_{j+1}(S(\mathbf{c}')) &= \sum_{\mathbf{b}^{j+1}|\mathcal{F}([\mathbf{h},\mathbf{b}^{j+1},\mathbf{0},\mathbf{0},\mathbf{0}])=\mathbf{c}'}P(\mathbf{b}^{j+1})P(\mathbf{y}_b^{j+1}|\mathbf{b}^{j+1})\\ &= P(b_{j+1}=0)P(y_{b_{j+1}}|b_{j+1}=0)\sum_{\mathbf{b}^j|\mathcal{F}([\mathbf{h},\mathbf{b}^j,\mathbf{0},\mathbf{0},\mathbf{0}])=\mathbf{c}'}P(\mathbf{b}^j)P(\mathbf{y}_b^j|\mathbf{b}^j)\\ &+ P(b_{j+1}=1)P(y_{b_{j+1}}|b_{j+1}=1)\sum_{\mathbf{b}^j|\mathcal{F}([\mathbf{h},\mathbf{b}^j,\mathbf{0},\mathbf{0},\mathbf{0}])=\mathbf{c}'\oplus\boldsymbol{\pi}(b_{j+1})}P(\mathbf{b}^j)P(\mathbf{y}_b^j|\mathbf{b}^j)\\ &= P(b_{j+1}=0)P(y_{b_{j+1}}|b_{j+1}=0)\cdot\alpha_j(S(\mathbf{c}'))\\ &+ P(b_{j+1}=1)P(y_{b_{j+1}}|b_{j+1}=1)\cdot\alpha_j(S(\mathbf{c}'\oplus\boldsymbol{\pi}(b_{j+1}))),\end{aligned}\tag{3.25}$$

avec les conditions d'initialisation (à l'instant $j=0$)

$$\alpha_0(S(\mathbf{c}'))=\begin{cases}1 & \text{pour } \mathbf{c}'=\mathcal{F}([\mathbf{h},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}])\\ 0 & \text{pour tout } \mathbf{c}'\neq\mathcal{F}([\mathbf{h},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}])\end{cases}.\tag{3.26}$$

L'équation finale dans (3.25) est la clef pour calculer $\alpha(S(\mathbf{c}'))$ par une méthode récursive (dans le sens direct) sur les bits de $\mathbf{b}$. Après $\ell(\mathbf{b})$ itérations, $\alpha_{\ell(\mathbf{b})}(S(\mathbf{c}'))=\alpha(S(\mathbf{c}'))$ dans (3.21).

De manière similaire, nous allons prouver que $\beta(S(\mathbf{c}''))$ dans (3.22) peut être déterminé récursivement sur les $\ell(\mathbf{r})$ bits de $\mathbf{r}$, pour tout $\mathbf{c}''\in GF(2)^{\ell(\mathbf{c})}$. Pour cela, nous considérons que $\bar{\mathbf{r}}^j=[0\dots0,r_{j+1}\dots r_{\ell(\mathbf{r})}]$ et $\mathbf{y}_{\bar{r}}^j=[0\dots0,y_{r_{j+1}}\dots y_{r_{\ell(\mathbf{r})}}]$. De plus, nous définissons

$$\beta_j(S(\mathbf{c}''))=\sum_{\bar{\mathbf{r}}^j}P(\bar{\mathbf{r}}^j)P(\mathbf{y}_{\bar{r}}^j|\bar{\mathbf{r}}^j)P(\mathbf{y}_c|\mathbf{c}''\oplus\mathcal{F}(\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\bar{\mathbf{r}}^j)),\tag{3.27}$$

comme la probabilité associée à l'état $S(\mathbf{c}'') \in \{0 \dots 2^{\ell(\mathbf{c})} - 1\}$ à l'instant $j \in \{0 \dots \ell(\mathbf{r})\}$ dans le treillis sur $\mathbf{r}$.

En appliquant (3.27) à l'état $S(\mathbf{c}'')$ à l'instant $j-1$, nous obtenons

$$\begin{aligned}
\beta_{j-1}(S(\mathbf{c}'')) &= \sum_{\bar{\mathbf{r}}^{j-1}} P(\bar{\mathbf{r}}^{j-1}) P(\mathbf{y}_{\bar{r}}^{j-1} | \bar{\mathbf{r}}^{j-1}) P(\mathbf{y}_c | \mathbf{c}'' \oplus \mathcal{F}([\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \bar{\mathbf{r}}^{j-1}])) \\
&= P(r_j = 0) P(y_{r_j} | r_j = 0) \sum_{\bar{\mathbf{r}}^{j}} P(\bar{\mathbf{r}}^{j}) P(\mathbf{y}_{\bar{r}}^{j} | \bar{\mathbf{r}}^{j}) P(\mathbf{y}_c | \mathbf{c}'' \oplus \mathcal{F}(\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \bar{\mathbf{r}}^{j})) \\
&+ P(r_j = 1) P(y_{r_j} | r_j = 1) \sum_{\bar{\mathbf{r}}^{j}} P(\bar{\mathbf{r}}^{j}) P(\mathbf{y}_{\bar{r}}^{j} | \bar{\mathbf{r}}^{j}) P(\mathbf{y}_c | \mathbf{c}'' \oplus \boldsymbol{\pi}(r_j) \oplus \mathcal{F}(\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \bar{\mathbf{r}}^{j})) \\
&= P(r_j = 0) P(y_{r_j} | r_j = 0) \cdot \beta_j(\mathbf{c}'') \\
&+ P(r_j = 1) P(y_{r_j} | r_j = 1) \cdot \beta_j(\mathbf{c}'' \oplus \boldsymbol{\pi}(r_j)).
\end{aligned} \tag{3.28}$$

A l'initialisation, c'est-à-dire quand $j = \ell(\mathbf{r})$,

$$\beta_{\ell(\mathbf{r})}(S(\mathbf{c}'')) = P(\mathbf{y}_c | \mathbf{c}''), \text{ pour tout } \mathbf{c}'' \in GF(2)^{\ell(\mathbf{c})}. \tag{3.29}$$

L'équation finale dans (3.28) est la clef pour calculer $\beta(S(\mathbf{c}''))$ par une méthode récursive (dans le sens indirect) sur les bits de $\mathbf{r}$. Après $\ell(\mathbf{r})$ itérations, $\beta_0(S(\mathbf{c}'')) = \beta(S(\mathbf{c}''))$ dans (3.22).

En étudiant les équations récursives introduites dans (3.25) and (3.28), il peut être facilement vérifié que les coefficients $\alpha_j(S(\mathbf{c}'))$ et $\beta_j(S(\mathbf{c}''))$ correspondent exactement aux coefficients définis dans [65] présentant l'algorithme BCJR. Plus précisément, les coefficients $\alpha(S(\mathbf{c}'))$ peuvent être calculés en réalisant une étape partielle dans le sens direct de l'algorithme BCJR (en commençant par le premier bit de $\mathbf{b}$ et en finissant par le dernier bit de $\mathbf{b}$). De manière similaire, les coefficients $\beta(S(\mathbf{c}''))$ peuvent être calculés en effectuant une étape partielle dans le sens indirect de l'algorithme BCJR (en commençant par le dernier bit de $\mathbf{r}$ et en finissant par le premier bit de $\mathbf{r}$). Les étapes pour déterminer la métrique globale (3.23), pour toutes les valeurs de $[\mathbf{u}, \mathbf{s}] \in \Omega_{[u,s]}$, avec la méthode présentée sont résumées ci-dessous :

Etape 1 : Initialiser $\alpha_0(S(\mathbf{c}'))$ et $\beta_{\ell(\mathbf{r})}(S(\mathbf{c}''))$ en respectant (3.26) et (3.29).
Etape 2 : Déterminer $\alpha_j(S(\mathbf{c}'))$, pour tout $\mathbf{c}' \in GF(2)^{\ell(\mathbf{c})}$ et pour tout $j = 1 \dots \ell(\mathbf{b})$, en utilisant (3.25) dans le sens direct (étape partielle de l'algorithme BCJR dans le sens direct).
Etape 3 : Déterminer $\beta_j(S(\mathbf{c}''))$, pour tout $\mathbf{c}'' \in GF(2)^{\ell(\mathbf{c})}$ et pour tout $j = \ell(\mathbf{r}) - 1 \dots 0$, en utilisant (3.28) dans le sens indirect (étape partielle de l'algorithme BCJR dans le sens indirect).
Etape 4 : Pour chaque $[\mathbf{u}, \mathbf{s}] \in \Omega_{[u,s]}$, calculer la métrique $\mathcal{M}([\mathbf{u}, \mathbf{s}])$ en utilisant (3.23), en gardant à l'esprit que $\alpha(S(\mathbf{c}')) = \alpha_{\ell(\mathbf{b})}(S(\mathbf{c}'))$ et $\beta(S(\mathbf{c}'')) = \beta_0(S(\mathbf{c}''))$.

Exemple 1 *Le code systématique* $(9, 7)$*, dont le polynôme générateur* $g(z) = z^2 + z + 1$*,*

possède une matrice de parité

$$\mathbf{\Pi} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

Dans cet exemple, nous considérons que l'en-tête connu $\mathbf{h} = [1]$ *et nous voulons déterminer la métrique* $\mathcal{M}([\mathbf{u}, \mathbf{s}])$ *pour la valeur de* $[\mathbf{u}, \mathbf{s}] = [0, 1]$*. Par conséquent,* $\ell(\mathbf{h}) + \ell(\mathbf{u}) + \ell(\mathbf{s}) = 3$ *bits et nous considérons que* $\ell(\mathbf{b}) = \ell(\mathbf{r}) = 2$ *bits. La modélisation du problème, suivant la méthode décrite dans cette partie, est illustrée sur la figure 3.12. Nous pouvons constater que* $\mathbf{h}$ *fixe l'état de départ du treillis sur* $\mathbf{b}$ *et que les vraisemblances du CRC initialisent les états finaux du treillis sur* $\mathbf{r}$*. Par ailleurs, les transitions pondérées par la valeur de* $[\mathbf{u}, \mathbf{s}]$ *permettent de connecter les deux treillis. La métrique* $\mathcal{M}([\mathbf{u}, \mathbf{s}])$ *est déterminée en sommant les probabilités associées aux différents chemins (dépendants de la valeur courante de* $[\mathbf{u}, \mathbf{s}]$*) connectant le treillis sur* $\mathbf{b}$ *avec le treillis sur* $\mathbf{r}$*.*

Avec cette méthode, une étape complète de décodage séquentiel (pour tous les $[\mathbf{u}, \mathbf{s}] \in \Omega_{[u,s]}$) est réalisée avec une complexité $\mathcal{O}((\ell(\mathbf{b}) + \ell(\mathbf{r}) + |\Omega_{[u,s]}|)2^{\ell(\mathbf{c})})$, comparée à $\mathcal{O}\left(|\Omega_{[u,s]}|2^{\ell(\mathbf{b})+\ell(\mathbf{r})}\right)$ avec la méthode basique (sans treillis).

3.5.2 Calcul approché de la métrique

En pratique, la plupart des CRCs ont une longueur supérieure à 16 bits et la complexité $\mathcal{O}(2^{\ell(\mathbf{c})})$ est encore trop importante pour permettre une implémentation en temps-réel de la méthode présentée dans la partie 3.5.1. Un calcul approché consiste à découper le CRC en M_p partitions of $\ell(\mathbf{c})/M_p$ bits, chaque partition étant considérée statistiquement indépendante des autres. Ce concept a déjà été utilisé dans [69] pour le décodage turbo des codes en bloc. Dans leur travail, les auteurs proposent un schéma de décodage itératif basé sur une segmentation des bits de parité. Dans notre cas, ce découpage est appliqué au CRC. Ainsi, $\mathbf{y}_c$ peut être décomposé tel que $\mathbf{y}_c = [\mathbf{y}_{c_1} \dots \mathbf{y}_{c_{M_p}}]$. En utilisant cette hypothèse d'indépendance, la somme de (3.18) devient

$$\Phi \approx \Phi_1 \cdot \Phi_2 \cdots \Phi_{M_p} = \prod_{m=1}^{M_p} \Phi_m, \tag{3.30}$$

avec

$$\Phi_m = \sum_{\mathbf{b},\mathbf{r}} P(\mathbf{b}) P(\mathbf{y}_b|\mathbf{b}) P(\mathbf{r}) P(\mathbf{y}_r|\mathbf{r}) P(\mathbf{y}_{c_m}|\mathcal{F}_m([\mathbf{h}, \mathbf{b}, \mathbf{u}, \mathbf{s}, \mathbf{r}])), \tag{3.31}$$

où $\mathcal{F}_m$ représente une fonction d'encodage associée aux colonnes allant de $(m-1) \cdot \frac{\ell(\mathbf{c})}{M_p} + 1$ à $m \cdot \frac{\ell(\mathbf{c})}{M_p}$ dans $\mathbf{\Pi}$. Cette fonction fournit un CRC partiel de $\ell(\mathbf{c})/M_p$ bits. La méthode pour

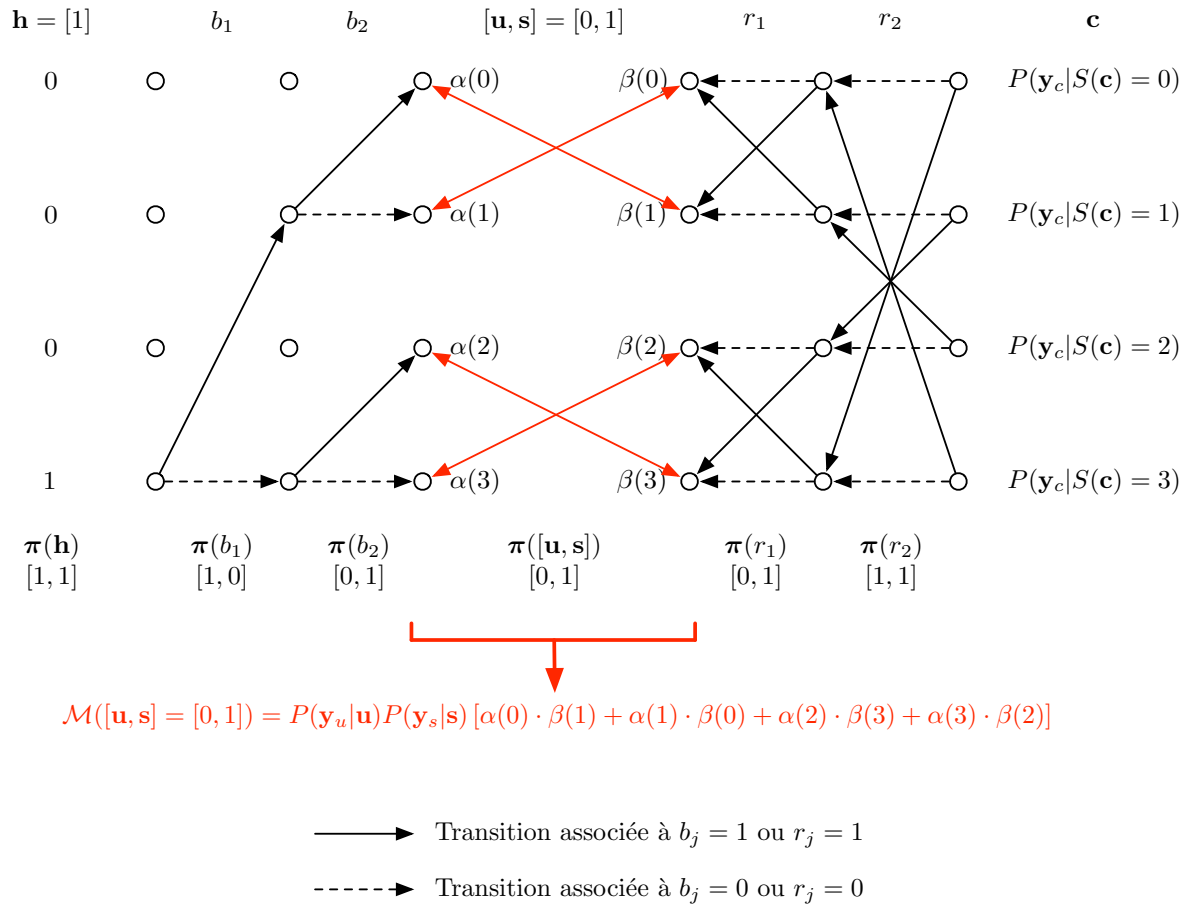


FIG. 3.12 – Illustration du calcul de la métrique à partir du treillis

déterminer Φ_m dans (3.31) est identique à celle utilisée pour calculer Φ dans la partie 3.5.1. La seule différence est associée à la taille du treillis : $2^{\ell(\mathbf{c})/M_p}$ états doivent être considérés à la place des $2^{\ell(\mathbf{c})}$ avec la méthode exacte (sans découper le CRC).

Avec cette hypothèse, la métrique globale définie dans (3.23) devient

$$\mathcal{M}([\mathbf{u}, \mathbf{s}]) = P(\mathbf{y}_u|\mathbf{u})P(\mathbf{y}_s|\mathbf{s}) \prod_{m=1}^{M_p} \sum_{\mathbf{c}'_m, \mathbf{c}''_m | \mathbf{c}''_m = \mathbf{c}'_m \oplus \mathcal{F}_m([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{0}])} \alpha^m(S(\mathbf{c}'_m)) \cdot \beta^m(S(\mathbf{c}''_m)), \quad (3.32)$$

Dans (3.32), $\alpha^m(S(\mathbf{c}'_m)) = \alpha^m_{\ell(\mathbf{b})}(S(\mathbf{c}'_m))$ et correspond à la probabilité associée à l'état $S(\mathbf{c}'_m)$ à l'instant $j = \ell(\mathbf{b})$ dans le m-ième treillis sur $\mathbf{b}$. De même, $\beta^m(S(\mathbf{c}''_m)) = \beta^m_0(S(\mathbf{c}''_m))$ et correspond à la probabilité associée à l'état $S(\mathbf{c}''_m)$ à l'instant $j = 0$ dans le m-ième treillis sur $\mathbf{r}$.

La complexité totale pour déterminer (3.32) est maintenant $\mathcal{O}((\ell(\mathbf{b})+\ell(\mathbf{r})+|\Omega_{[u,s]}|)M_p 2^{\ell(\mathbf{c})/M_p})$, au prix d'une performance faiblement sous-optimale.

3.6 Résultats de simulation

Dans le profil étendu, des outils robustes sont fournis par la norme H.264/AVC. Le partitionnement des données, présenté dans la partie 1.3.4, permet de classer les données des *slices* comprimés en fonction de leur influence sur la qualité de la vidéo décodée. Dans cette technique, les données sont réparties en trois partitions DPA, DPB et DPC.

Cette décomposition du flux permet d'adapter le niveau de protection des partitions en fonction de leur sensibilité. Chaque partition est encapsulée dans un paquet NAL qui est ensuite transmis sur le réseau. Au niveau des couches inférieures, les paquets associés à la partition A sont fortement protégés et nous considérons que le récepteur les reçoit correctement. En revanche, les paquets contenant DPB et DPC sont transmis sur des canaux bruités et des erreurs peuvent survenir durant la transmission. Rappelons simplement que DPB et DPC contiennent les résidus de prédiction du *slice*, encodés avec la méthode CAVLC (expliquée dans la partie 1.3.2).

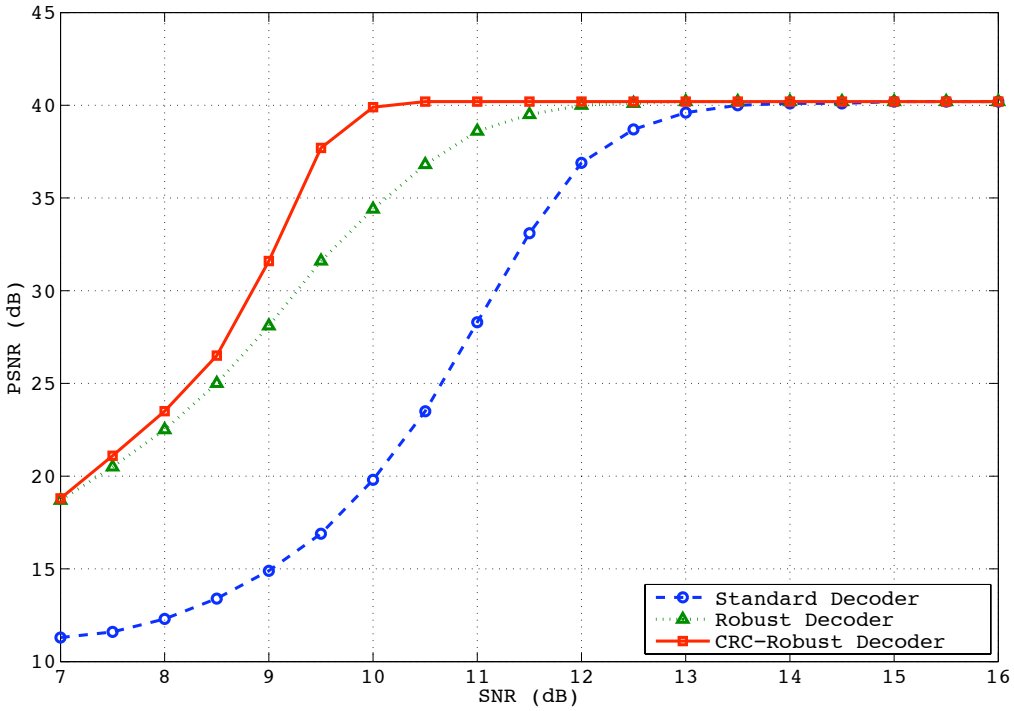


FIG. 3.13 – PSNR de la vidéo décodée en fonction du SNR lors d'une transmission sans codage canal

La méthode robuste présentée dans ce chapitre a été appliquée au décodage des séquences CAVLC contenues dans les partitions B et C. Chaque séquence CAVLC est considérée comme un groupe indépendant de mots de code qui peut être séparé des autres grâce aux marqueurs de synchronisation. Par conséquent, la méthode de décodage séquentiel basée sur les

groupes (présentée dans la partie 3.4) est préférée à la méthode générale décrite dans la partie 3.3. Précisons juste que les séquences CAVLC du H.264/AVC ne sont pas totalement indépendantes (voir annexe A pour plus de détails), mais les faibles dépendances existantes peuvent être négligées. Lors des simulations, les performances de la méthode présentée ont été comparées à deux autres méthodes de décodage : une méthode de décodage standard et une méthode de décodage robuste exploitant seulement les propriétés structurelles.

Le système de transmission est composé d'un serveur, d'un point d'accès WiFi, d'un canal radio et d'un terminal WiFi. Le serveur encode de manière répétitive les 5 premières images de *Foreman.cif* dans un GOP IPPPP en utilisant un codeur H.264/AVC travaillant en mode CAVLC. Il génère les partitions respectives (paquets vidéo) qui sont ensuite transmises sur le réseau coeur. Ce réseau coeur est considéré idéal et tous les paquets arrivent sans encombre au point d'accès WiFi.

A la couche MAC du point d'accès, les paquets IP arrivant du réseau coeur sont répartis sur plusieurs fragments MAC de taille fixe. Un CRC de 4 octets, respectant le protocole MAC 802.11 défini dans la partie 2.3.2, est ajouté à la fin de chaque fragment. Au niveau de la couche PHY, les données sont encodées à l'aide du codeur convolutif de la norme 802.11*a* (introduit dans la partie 2.3.1). Elles sont finalement modulées en BPSK avant d'être transmises sur le canal radio.

Pour améliorer les performances du décodage, les marqueurs de synchronisation, mentionnés dans la partie 3.4, sont insérés dans le flux transmis. Rappelons simplement que ces marqueurs spécifient la longueur de chaque séquence CAVLC contenue dans les partitions B et C. Ces informations sont transmises dans des paquets NAL spécifiques et chaque paramètre de longueur est encodé en Exp-Golomb (voir partie 1.3.2). Cette méthode d'encapsulation permet de rester compatible avec la norme H.264/AVC : un décodeur standard ignorera les paquets NAL contenant les marqueurs de synchronisation mais pourra décoder le flux vidéo sans exploiter ces informations. Il faut cependant noter que le surcoût de débit, associé à la transmission de ces redondances, représente environ 30 % du débit total. Dans ces simulations, nous considérons que les paquets contenant les marqueurs de synchronisation, tout comme les partitions A, ne sont pas dégradés par le canal radio. Seules les données contenues dans les partitions B et C sont bruitées durant la transmission hertzienne. La valeur du SNR (*Signal to Noise Ratio*) permet de modifier la variance du bruit blanc gaussien sur le canal radio (le canal est considéré AWGN).

Au niveau du récepteur, les données sont décodées à la couche PHY par un algorithme BCJR [65] (décodage canal SISO) et sont envoyées à la couche APL en utilisant le mécanisme perméable présenté dans la partie 3.2. Comme nous l'avons énoncé précédemment, trois types de décodeurs sont considérés dans la couche APL :

1. Un décodeur *standard* qui réalise des décisions dures sur les données souples reçues en tirant parti des marqueurs de synchronisation.
2. Un décodeur *robust* qui exploite à la fois les données souples, les propriétés structurelles du flux et les marqueurs de synchronisation, en négligeant cependant les redondances fournies par le CRC. Ce décodeur utilise l'algorithme défini dans la partie 3.4, mais la

métrique ne contient pas le terme Φ.

3. Un décodeur *CRC-robust* qui combine toutes les sources de redondances précédentes ainsi que les propriétés du CRC. Ce décodeur utilise la méthode définie dans la partie 3.4.

Dans nos simulations, $M = 20$ et $\ell(\mathbf{s}) = 4$ bits par défaut, pour les deux décodeurs robustes. Par ailleurs, le décodeur robuste exploitant le CRC utilise la méthode sous-optimale présentée dans la partie 3.5.2. Dans notre cas, le CRC est découpé en 4 partitions de 8 bits.

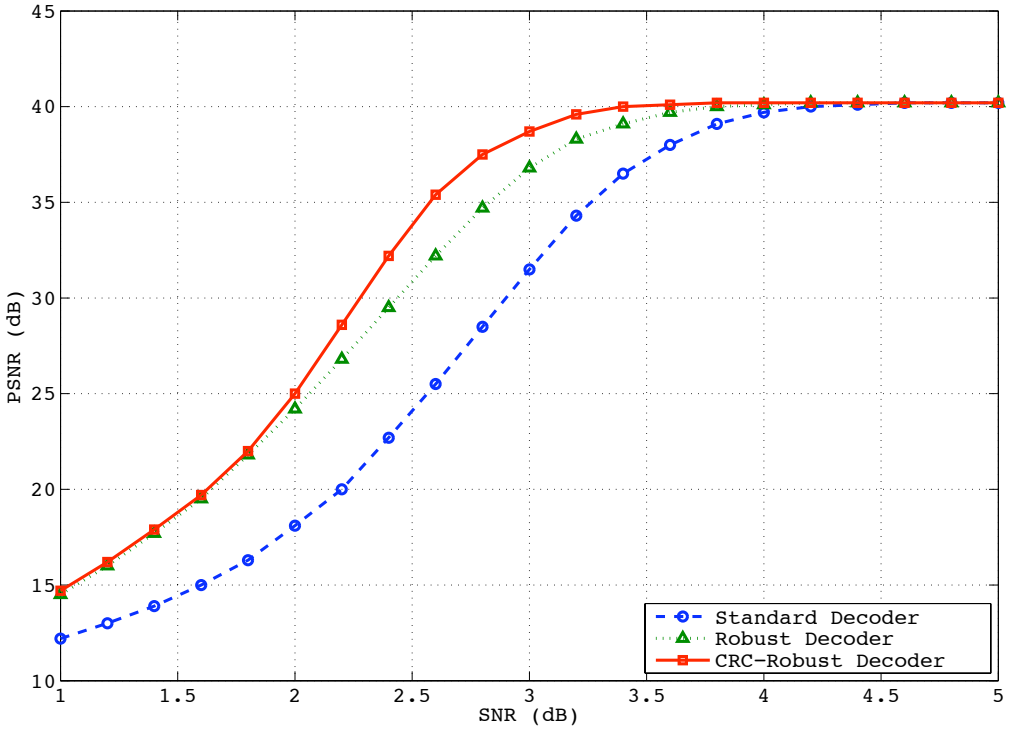


FIG. 3.14 – PSNR de la vidéo décodée en fonction du SNR lors d'une transmission avec codage canal

Les figures 3.14 et 3.13 montrent l'évolution du PSNR (*Peak Signal to Noise Ratio*) de la vidéo décodée en fonction du SNR pour les trois types de décodeurs, *avec et sans* codage canal respectivement. Lors des simulations associées à la figure 3.13, le codage/décodage canal de la couche PHY a été désactivé. Les données sont donc simplement modulées en BPSK dans la couche PHY avant d'être transmises sur le canal AWGN. Sur les deux figures, les décodeurs *standard*, *robust* et *CRC-robust* sont comparés pour une taille de *payload* MAC de 100 octets. Dans tous les cas, nous pouvons remarquer que les performances du décodeur standard sont dépassées par celles des deux décodeurs robustes. Par ailleurs, les deux décodeurs robustes ont des performances équivalentes pour des faibles valeurs de SNR. Ce n'est qu'à partir d'un certain seuil que le décodage *CRC-robust* commence à prendre l'avantage sur le décodage

robust. Au dessus de ce seuil, le gain de codage augmente avec le SNR. Ce comportement est typique dans le domaine du décodage canal. Dans nos simulations, la valeur du seuil est d'environ 8.5 dB pour la figure 3.13 et d'environ 1.8 dB pour la figure 3.14.

FIG. 3.15 – Qualité de la 5-ième image de *Foreman.cif* obtenue après (a) un décodage sans erreur, (b) un décodage *standard*, (c) un décodage *robust*, et (d) un décodage *CRC-robust* pour un SNR de 2.8 dB, lors d'une transmission avec codage canal

La figure 3.15 expose la 5-ième image originale de la séquence vidéo *Foreman.cif*, ainsi que ses reproductions obtenues après transmission et décodage. Dans ce cas, le codage/décodage canal de la couche PHY a été activé. Par ailleurs, ces résultats ont été obtenus pour des tailles de *payload* MAC de 100 octets et pour un SNR de 2.8 dB pour lequel les valeurs de PSNR atteintes par les décodeurs *standard*, *robust* et *CRC-robust* sont respectivement de 29, 35 and 38 dB (voir figure 3.14). Nous pouvons constater que l'image générée par le décodeur *standard* contient de nombreux artefacts : sa qualité est donc très faible. Le décodeur *robust* permet d'améliorer considérablement la qualité vidéo, même si certains artefacts restent encore visibles. Finalement, aucune différence visuelle ne peut être distinguée entre l'image

originale et l'image obtenue avec le décodeur *CRC-robust*.

3.7 Conclusion

Dans ce chapitre, nous avons présenté un estimateur au sens MAP pour le décodage robuste de la vidéo. Ce décodeur exploite conjointement les propriétés sémantiques et syntaxiques du flux vidéo encodé et le CRC des fragments de la couche MAC. Nous avons démontré que l'implémentation pratique de cette technique pouvait être basée sur la combinaison d'un algorithme de décodage séquentiel et d'un algorithme BCJR. Nous avons appliqué cette méthode au décodage des résidus de prédiction du H.264/AVC. Les résultats de simulation ont montré que les redondances introduites par le CRC améliorent l'efficacité du décodage robuste. Plus précisément, nous avons pu constater que l'utilisation conjointe des propriétés structurelles du flux et du CRC devenait intéressante à partir d'une certaine valeur de SNR. Pour faciliter et renforcer le décodage, des marqueurs de synchronisation ont été insérés dans le flux transmis. Ces marqueurs spécifient la taille de chaque séquence CAVLC et représentent actuellement un sur-débit important dans les expériences présentées. L'objectif futur sera de réduire ce débit complémentaire en ne transmettant, par exemple, que la taille associée aux séquences CAVLC contenues dans un macrobloc complet.

Chapitre 4

Implantation du décodage robuste

Dans ce chapitre, nous présentons le deuxième travail abordé durant cette thèse.

4.1 Position du problème

Comme nous l'avons vu au chapitre 2, la transmission paquétisée d'informations multimédia s'appuie souvent sur la pile protocolaire du type IP/UDP/RTP. La figure 4.1 illustre les procédures de réassemblage des paquets à travers les couches protocolaires d'un récepteur WiFi. Des mécanismes de détection d'erreurs sont implémentés dans chaque couche, ils sont listés ci-dessous :

- Au niveau de la couche PHY (voir partie 2.3.1), un CRC de 2 octets protège les champs de l'en-tête. Les paquets contenant des en-têtes détériorés sont effacés.
- A la couche MAC (voir partie 2.3.2), un CRC de 4 octets protège l'en-tête et la *payload*. Quand une erreur survient, le fragment est retransmis par le point d'accès.
- A la couche IP (voir partie 2.3.3), les champs de l'en-tête sont protégés par un checksum de 2 octets. Les paquets contenant des en-têtes erronés sont considérés perdus.
- A la couche UDP (voir partie 2.3.4), un checksum de 2 octets protège la totalité du paquet. Quand une erreur intervient, le paquet est effacé.

Les mécanismes de détection d'erreurs basés sur les CRCs et les checksums, combinés au système de retransmission de la couche MAC, permettent à la couche APL de recevoir des paquets corrects. Néanmoins, ces outils entraînent une augmentation du délai de transmission et une consommation excessive de ressources radio car les retransmissions peuvent devenir très fréquentes quand les conditions radio se dégradent. Pour réduire ce délai, le nombre de retransmission est souvent limité. Cette limitation engendre des pertes de paquets au niveau de la couche APL. Or, chaque paquet contient une grande partie des informations visuelles et leur perte entraîne donc une détérioration significative de la qualité vidéo.

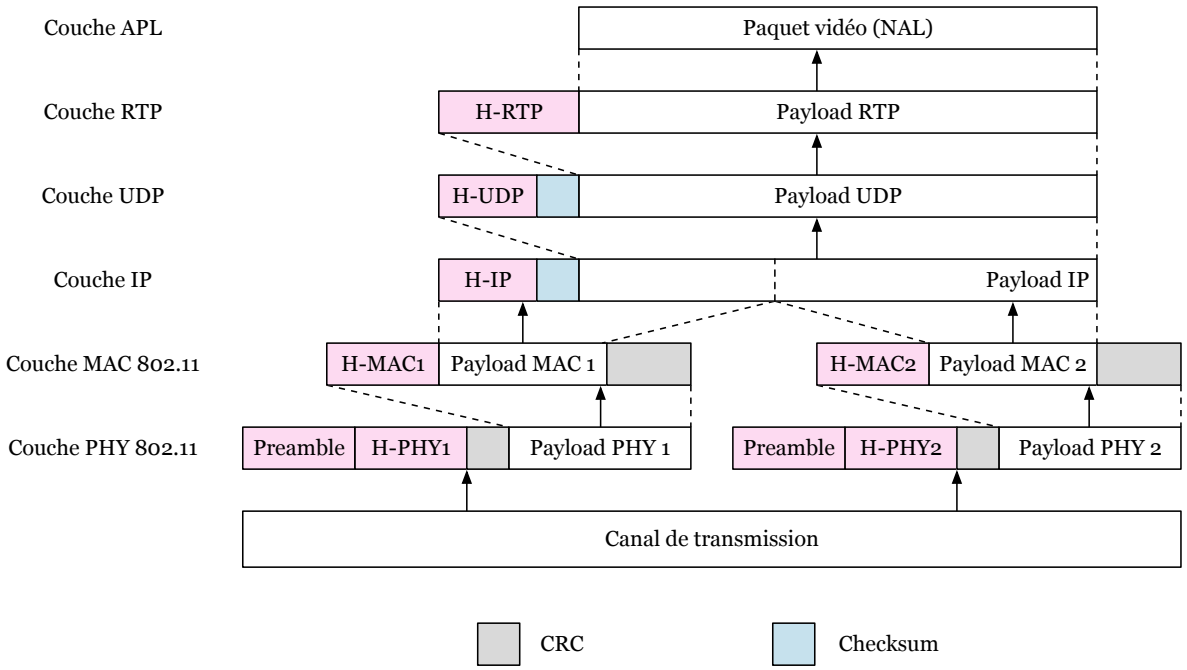


FIG. 4.1 – Mise en évidence des codes de détection d'erreurs

Ces dernières années, des techniques de décodage conjoint source-canal ont été proposées pour corriger les erreurs contenues dans les paquets vidéo reçus. Ces méthodes ont été largement décrites dans le chapitre 3. Retenons uniquement que ces mécanismes robustes exploitent les redondances inhérentes contenues dans les informations transmises pour corriger des erreurs. Ces redondances peuvent correspondre aux propriétés sémantiques et syntaxiques du codeur source, aux informations souples fournies par la couche PHY, aux propriétés de la paquétisation, etc. Toutes ces sources de redondances sont combinées au niveau du décodeur de la couche APL pour renforcer l'efficacité du décodage vidéo.

Cependant, les techniques JSCD ne sont pas compatibles avec le fonctionnement actuel de la pile protocolaire. Pour être efficaces, ces décodeurs robustes ont besoin de recevoir les informations souples issues de la couche PHY. Or, les paquets binaires erronés sont effacés par les mécanismes de détection d'erreurs avant d'avoir atteint le décodeur vidéo de la couche APL.

4.2 Nouveau modèle de couche perméable

Plusieurs modifications ont été proposées dans la littérature pour rendre la pile perméable. Le protocole UDP-Lite [70] correspond à une solution efficace pour la couche UDP. Dans ce schéma, le checksum ne protège qu'un nombre limité d'octets (généralement associés aux informations contenues dans les en-têtes UDP-Lite, RTP et APL). Par ailleurs, les travaux

présentés dans [71, 72] introduisent des mécanismes destinés à la couche Liaison. Ils proposent de désactiver la procédure de détection d'erreurs basée sur le CRC et considère que l'en-tête des fragments est correctement reçu. En utilisant un schéma incorporant ces deux mécanismes, toutes les données vidéo peuvent remonter au décodeur de la couche APL. En revanche, les paquets contenant des en-têtes erronés sont encore effacés (même si leur *payload* contient un nombre limité d'erreurs). Le point bloquant de cette méthode est donc ramené aux erreurs survenant dans les en-têtes.

Le modèle de couche présenté dans ce chapitre tente de résoudre ce problème. Il est basé sur la constatation suivante : les seules informations utilisées par une couche sont situées dans l'en-tête des paquets respectifs. En d'autres termes, les traitements réalisés par une couche manipulent uniquement les informations contenues dans l'en-tête des paquets et les données contenues dans la *payload* n'ont pas de signification à ce niveau. Lorsque les champs de l'en-tête sont corrects, la couche est capable de transmettre la *payload* à la couche supérieure. Dans notre modèle amélioré, nous cherchons donc à corriger les informations importantes, contenues dans l'en-tête des paquets, avant de les utiliser. L'outil fondamental de cette méthode exploite les différentes sources de redondances contenues dans la pile protocolaire pour corriger les en-têtes. De plus, ce mécanisme permet de remonter les informations souples, fournies par le décodage SISO de la couche PHY, jusqu'au décodeur de la couche APL.

La technique de correction des en-têtes fait intervenir deux sources de redondance :

1. Les redondances intra-couche et inter-couches introduites par les propriétés structurelles de la pile. Ce type de corrélation a déjà été exploité dans le protocole ROHC (*Robust Header Compression*) [73] dans lequel les en-têtes IP, UDP et RTP sont remplacés par une version comprimée. Les performances atteintes par ROHC sont significatives puisque la taille des en-têtes passe de 40 octets à seulement quelques octets. Dans notre étude, ces redondances sont utilisées pour établir les informations *a priori* sur les en-têtes. Elles permettent de construire un ensemble limité de combinaisons d'en-tête (candidats).
2. Les redondances introduites par les codes de détection d'erreurs (CRC ou checksum). Ces codes sont insérés dans pratiquement tous les paquets de la pile, comme l'illustre la figure 4.1. Dans notre étude, les CRCs et les checksums sont utilisés comme des codes de correction d'erreurs [63, 64]. Ils permettent de sélectionner la meilleure combinaison parmi l'ensemble des candidats.

Au niveau de chaque couche, le traitement de décodage des en-têtes combine les informations souples fournies par la couche inférieure, les redondances du CRC ou du checksum, ainsi que les informations *a priori* résultant des propriétés structurelles de la pile. Une fois l'en-tête corrigé, les champs utiles sont traités normalement par la couche. Les informations souples relatives à la *payload* sont ensuite transmises à la couche supérieure. La figure 4.2 illustre le principe de fonctionnement d'une telle couche. Elle met en avant son intérêt majeur : la propagation des informations souples entre la couche $L-1$ à la couche $L+1$. Elle représente également les diverses sources de redondance qui sont exploitées pour faciliter la correction des en-têtes à la couche L.

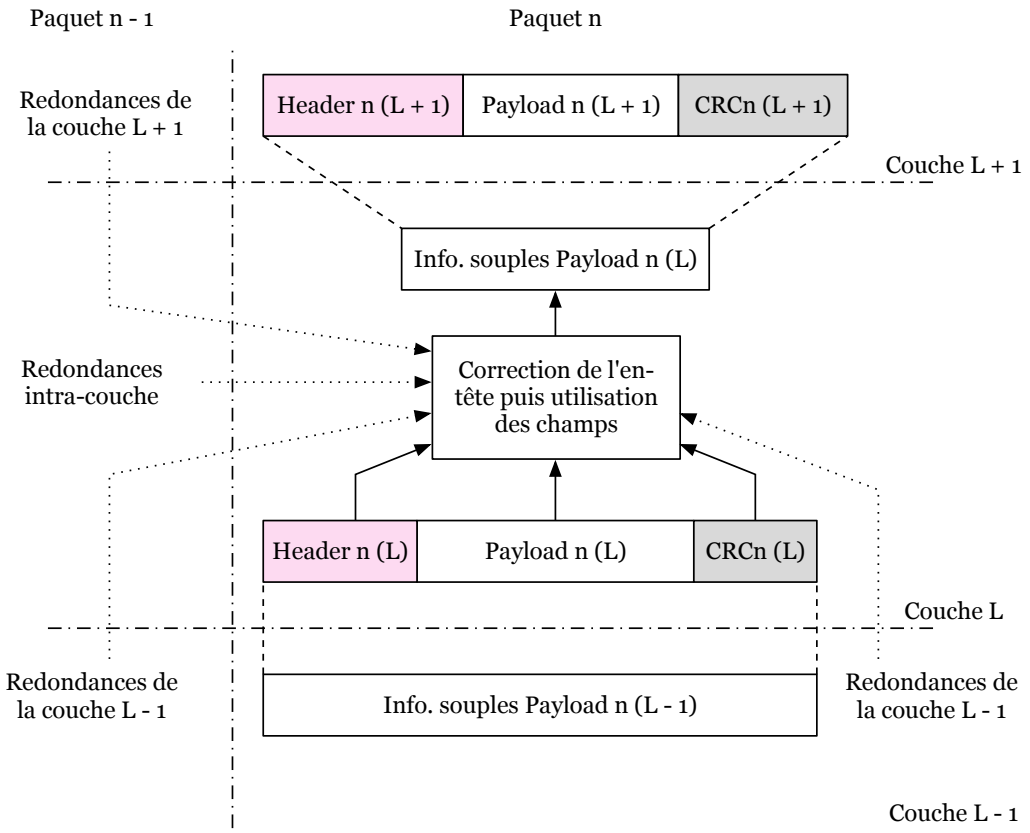


FIG. 4.2 – Principe de fonctionnement du nouveau modèle de couche perméable

Dans cette étude, nous focalisons sur les couches PHY et MAC du standard 802.11. Néanmoins, cette méthode peut être étendue à toutes les couches protocolaires.

L'organisation de ce chapitre suit le plan suivant : l'estimateur MAP général de correction des en-têtes est défini dans la partie 4.3. Dans la partie 4.4, nous discutons des problèmes de complexité liés au calcul de l'estimateur dans des cas pratiques. L'application du mécanisme aux couches PHY et MAC de WiFi est introduite dans la partie 4.5. Finalement, les résultats de simulation sont présentés dans la partie 4.6.

4.3 Estimateur MAP de décodage des en-têtes

De manière générale, le n-ième paquet arrivant au niveau d'une couche L est composé d'un en-tête, d'une *payload* et d'un CRC. Les informations protégées par le CRC $\mathbf{c}_n^L$ de $\ell(\mathbf{c}_n^L)$ bits peuvent être classées en quatre catégories :

1. Les champs constants, représentés par le vecteur $\mathbf{k}_n^L$ de $\ell(\mathbf{k}_n^L)$ bits, sont considérés connus (*known*). Les valeurs de ces champs ne varient pas dans les différents paquets.

2. Les champs déductibles (*predictable*) sont incorporés dans le vecteur $\mathbf{p}_n^L$ de $\ell(\mathbf{p}_n^L)$ bits. Contrairement aux champs connus, les champs déductibles peuvent être estimés en exploitant les redondances intra-couche et inter-couches représentées par R_n^L. Ils sont prédits en utilisant les informations contenues dans les paquets précédemment reçus. Il est donc évident que les champs déductibles sont parfaitement déterminés si les paquets précédents ont été correctement reçus.
3. Les champs importants inconnus (*unknown*) sont rassemblés dans le vecteur $\mathbf{u}_n^L$ de $\ell(\mathbf{u}_n^L)$ bits. Ces paramètres peuvent être totalement inconnus ou limités à un ensemble de combinaisons contenues dans $\Omega_u^{n,L}(\mathbf{k}_n^L, \mathbf{p}_n^L, R_n^L)$. Ces combinaisons sont déterminées à partir des valeurs de $\mathbf{k}_n^L$, $\mathbf{p}_n^L$ et R_n^L.
4. Le vecteur $\mathbf{o}_n^L$ de $\ell(\mathbf{o}_n^L)$ bits contient les autres informations (*other*) protégées par le CRC. Cette dernière catégorie regroupe les données inconnues qui ne sont pas nécessaires dans les traitements du paquet dans la couche L, mais qui peuvent être utiles dans la couche supérieure $(L+1)$.

De manière plus précise, R_n^L contient les informations contenues dans tous les en-têtes du paquet $n-1$ (au niveau des couches $L-1$, L et $L+1$) ainsi que celles de l'en-tête du paquet n à la couche $L-1$. Nous pouvons donc écrire que

$$R_n^L = \{\mathbf{k}_{n-1}^{L-1}, \mathbf{k}_{n-1}^{L}, \mathbf{k}_{n-1}^{L+1}, \mathbf{k}_{n}^{L-1}, \mathbf{p}_{n-1}^{L-1}, \mathbf{p}_{n-1}^{L}, \mathbf{p}_{n-1}^{L+1}, \mathbf{p}_{n}^{L-1}, \mathbf{u}_{n-1}^{L-1}, \mathbf{u}_{n-1}^{L}, \mathbf{u}_{n-1}^{L+1}, \mathbf{u}_{n}^{L-1}\}.$$

Par ailleurs, les données du paquet non protégées par le CRC au niveau de la couche L sont incluses dans $\mathbf{x}_n^L$.

Tous les bits protégés par le CRC sont collectés dans le vecteur $\mathbf{r}_n^L = [\mathbf{k}_n^L, \mathbf{p}_n^L, \mathbf{u}_n^L, \mathbf{o}_n^L]$ qui contient tous les champs mentionnés ci-dessus. Notons juste que l'ordre des bits de $\mathbf{r}_n^L$ ne correspond pas à l'ordre dans lequel les données sont transmises dans le n-ième paquet, nous utilisons cette notation pour simplifier le problème théorique. Le CRC $\mathbf{c}_n^L$ associé à $\mathbf{r}$ est déterminé par $\mathbf{c}_n^L = \mathcal{F}^L(\mathbf{r}_n^L)$, où $\mathcal{F}^L$ est une fonction d'encodage générique associée à la couche L.

Pour faciliter la lecture du document, les indices n et L ont été volontairement retirés des équations dans la suite des développements théoriques.

Comme nous l'avons montré précédemment, l'évaluation de $\mathbf{c}$ dépend d'un polynôme générateur $g(z) = \sum_{i=0}^{\ell(\mathbf{c})} a_i z^i$ caractérisant le CRC. Une matrice génératrice systématique $\mathbf{G} = [\mathbf{I}, \mathbf{\Pi}]$ peut être associée à $g(z)$. Le CRC $\mathbf{c}$ peut alors être déterminé en utilisant un traitement récursif sur $\ell(\mathbf{r})$ bits du vecteur $\mathbf{r}$ comme suit

$$\begin{cases} \mathbf{c}^0 = \mathbf{0}, \\ \mathbf{c}^{j+1} = \mathcal{F}(\mathbf{r}^{j+1}) = \mathbf{c}^j \oplus (r_{j+1} \cdot \boldsymbol{\pi}(r_{j+1})). \end{cases} \tag{4.1}$$

Dans (4.1), $\mathbf{r}^j = [r_1 \dots r_j, 0 \dots 0]$, $\boldsymbol{\pi}(r_j)$ représente le vecteur de parité (contenu dans $\mathbf{\Pi}$) associé au bit r_j et $\oplus$ définit l'opérateur XOR (*ou exclusif*). Après $\ell(\mathbf{r})$ itérations, $\mathbf{c}^{\ell(\mathbf{r})} = \mathcal{F}(\mathbf{r}) = \mathbf{c}$.

Considérons maintenant que les données ont été transmises sur un canal AWGN introduisant un bruit de moyenne nulle et de variance σ^2 et que les informations souples sont envoyées de couche en couche au niveau du récepteur. A l'entrée de la couche L, les données reçues sont notées $\mathbf{y} = [\mathbf{y}_k, \mathbf{y}_p, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c]$ et correspondent aux observations respectives de $\mathbf{k}$, $\mathbf{p}$, $\mathbf{u}$, $\mathbf{o}$ et $\mathbf{c}$.

Puisque $\mathbf{k}$ est connu et que $\mathbf{p}$ peut être totalement prédit, seul $\mathbf{u}$ reste à prédire. L'estimateur optimal $\widehat{\mathbf{u}}$ au sens MAP est donc donné par

$$\widehat{\mathbf{u}} = \arg\max_{\mathbf{u}} P(\mathbf{u}|\mathbf{k}, \mathbf{p}, R, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c). \tag{4.2}$$

Cet estimateur prend en compte les observations de $\mathbf{y}$, la connaissance de $\mathbf{k}$, $\mathbf{p}$ et R, ainsi que les propriétés du CRC. D'après Bayes, (4.2) peut être reformulé comme suit

$$\widehat{\mathbf{u}} = \arg\max_{\mathbf{u}} P(\mathbf{u}, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, R). \tag{4.3}$$

Dans (4.3), $P(\mathbf{u}, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, R)$ peut être exprimée comme suit

$$P(\mathbf{u}, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, R) = \sum_{\mathbf{o}} P(\mathbf{u}, \mathbf{o}, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, R). \tag{4.4}$$

D'après (4.4), nous pouvons remarquer que $P(\mathbf{u}, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, R)$ peut être déterminée en marginalisant $P(\mathbf{u}, \mathbf{o}, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, R)$ sur les $2^{\ell(\mathbf{o})}$ combinaisons de $\mathbf{o}$. Par ailleurs, en considérant que $\mathbf{o}$ et $\mathbf{c}$ sont indépendants de R et en gardant à l'esprit que le canal est sans mémoire, on peut écrire que

$$\begin{aligned} P(\mathbf{u}, \mathbf{o}, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, R) &= P(\mathbf{u}|\mathbf{k}, \mathbf{p}, R) P(\mathbf{y}_u|\mathbf{k}, \mathbf{p}, R, \mathbf{u}) P(\mathbf{o}, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, R, \mathbf{u}, \mathbf{y}_u) \\ &= P(\mathbf{u}|\mathbf{k}, \mathbf{p}, R) P(\mathbf{y}_u|\mathbf{u}) P(\mathbf{o}, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, \mathbf{u}). \end{aligned} \tag{4.5}$$

En combinant (4.4) et (4.5), nous obtenons

$$\begin{aligned} P(\mathbf{u}, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, R) &= P(\mathbf{u}|\mathbf{k}, \mathbf{p}, R) P(\mathbf{y}_u|\mathbf{u}) \sum_{\mathbf{o}} P(\mathbf{o}, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, \mathbf{u}) \\ &= P(\mathbf{u}|\mathbf{k}, \mathbf{p}, R) P(\mathbf{y}_u|\mathbf{u}) \Psi(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_c), \end{aligned} \tag{4.6}$$

avec

$$\Psi(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_c) = \sum_{\mathbf{o}} P(\mathbf{o}, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, \mathbf{u}). \tag{4.7}$$

Dans (4.6), $P(\mathbf{u}|\mathbf{k}, \mathbf{p}, R)$ représente la probabilité *a priori* de $\mathbf{u}$ connaissant $\mathbf{k}$, $\mathbf{p}$ et R. De manière générale, nous considérons que les valeurs possibles de $\mathbf{u}$ sont rassemblées dans $\Omega_u = \Omega_u(\mathbf{k}, \mathbf{p}, R)$. En supposant que les combinaisons possibles de $\mathbf{u}$ (contenues dans Ω_u) sont équiprobables, nous pouvons écrire que

$$P(\mathbf{u}|\mathbf{k}, \mathbf{p}, R) = P(\mathbf{u}|\Omega_u) = \frac{1}{|\Omega_u|},$$

où $|\Omega_u|$ représente le cardinal de l'ensemble Ω_u. Par conséquent, en intégrant (4.6) dans (4.3), nous aboutissons à

$$\widehat{\mathbf{u}} = \arg\max_{\mathbf{u} \in \Omega_u} P(\mathbf{y}_u|\mathbf{u}) \Psi(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_c). \tag{4.8}$$

Dans (4.8), $P(\mathbf{y}_u|\mathbf{u})$ représente la vraisemblance de $\mathbf{u}$ et $\Psi(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_c)$ correspond à une somme dont la complexité est exponentielle en $\ell(\mathbf{o})$.

4.4 Evaluation de l'estimateur dans un cas pratique

L'estimateur défini dans (4.8) peut être employé dans un contexte très général. Dans cette partie, nous nous intéressons à la détermination de cet estimateur dans différentes situations.

4.4.1 Quand l'ensemble o n'existe pas

Il existe de nombreuses situations dans lesquelles les bits protégés par le CRC appartiennent seulement aux ensembles $\mathbf{k}$, $\mathbf{p}$ et $\mathbf{u}$. Dans ces cas, $\mathbf{o}$ n'existe pas et (4.8) devient

$$\widehat{\mathbf{u}} = \arg\max_{\mathbf{u}\in\Omega_u} P(\mathbf{y}_u|\mathbf{u})P(\mathbf{y}_c|\mathcal{F}([\mathbf{k},\mathbf{p},\mathbf{u}])), \tag{4.9}$$

où $\mathcal{F}([\mathbf{k},\mathbf{p},\mathbf{u}])$ peut être directement déterminé grâce au schéma récursif défini dans (4.1). Ainsi, un simple calcul de CRC remplace la somme sur toutes les combinaisons possibles de $\mathbf{o}$ et la complexité de l'estimation devient négligeable.

4.4.2 Quand l'ensemble o existe

Pour faciliter la lecture du document, $\Psi(\mathbf{k},\mathbf{p},\mathbf{u},\mathbf{y}_o,\mathbf{y}_c)$ est remplacé par Ψ dans la suite de cette partie.

Quand $\mathbf{o}$ existe, le terme Ψ dans (4.8) devient complexe à déterminer. En considérant que les bits de $\mathbf{o}$ sont i.i.d. et ne dépendent pas des autres paramètres, (4.7) devient

$$\Psi = \sum_{\mathbf{o}} P(\mathbf{o})P(\mathbf{y}_o|\mathbf{o})P(\mathbf{y}_c|\mathcal{F}([\mathbf{k},\mathbf{p},\mathbf{u},\mathbf{o}])). \tag{4.10}$$

La détermination de (4.10) consiste donc à sommer les probabilités associées aux $2^{\ell(\mathbf{o})}$ combinaisons de $\mathbf{o}$ et à leurs CRC respectifs. Il est évident qu'un calcul direct possède une complexité exponentielle en $\ell(\mathbf{o})$. Dans cette partie, nous présentons deux méthodes de complexité réduite qui sont utiles pour calculer l'estimateur (4.8) : la première fournit une valeur exacte de Ψ tandis que la seconde évalue une valeur approchée de Ψ (au prix d'une complexité beaucoup plus faible).

Calcul exact de Ψ

Comme nous l'avons démontré dans (4.1), le CRC peut être calculé récursivement sur les bits de $\mathbf{r}$. La valeur du CRC associée aux $j+1$ premiers bits de $\mathbf{r}$ dépend seulement de la valeur du CRC à l'instant j et du $j+1$-ième bit de $\mathbf{r}$. Chaque valeur de CRC à l'instant j conduit à deux valeurs de CRC à l'instant $j+1$. Par conséquent, l'évolution des valeurs du CRC en fonction des bits de $\mathbf{r}$ peut être décrite par un treillis. Dans ce treillis, les états

correspondent aux $2^{\ell(\mathbf{c})}$ valeurs possibles de CRC et les transitions sont déterminées par les bits de $\mathbf{r}$. A chaque instant j, nous étudions la contribution de r_j (j-ième bit de $\mathbf{r}$) sur le CRC global.

Dans notre cas, $\mathbf{r} = [\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{o}]$. Les données contenues dans $\mathbf{k}$ et $\mathbf{p}$ sont considérées connues et nous cherchons à trouver la meilleure combinaison de $\mathbf{u} \in \Omega_u$ en tenant compte des redondances introduites par le CRC $\mathbf{c}$. Le treillis est donc appliqué aux portions $\mathbf{o}$ et $\mathbf{c}$ pour des valeurs fixes de $\mathbf{k}$, $\mathbf{p}$ et $\mathbf{u}$. Les états représentent les valeurs possibles du CRC et le treillis regroupe les combinaisons de $\mathbf{o}$ qui donnent les mêmes valeurs de CRC.

Dans la suite, nous considérons que l'état associé à une valeur possible de CRC $\mathbf{c}'$ est noté $S(\mathbf{c}')$, $\mathbf{c}'$ étant la représentation binaire de $S(\mathbf{c}') \in \{0 \dots 2^{\ell(\mathbf{c})} - 1\}$. En partant de (4.10), Ψ peut être reformulé tel que

$$\begin{aligned} \Psi &= \sum_{\mathbf{o}} P(\mathbf{o}) P(\mathbf{y}_o|\mathbf{o}) P(\mathcal{F}([\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{0}]) \oplus \mathcal{F}([\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{o}])) \\ &= \beta(S(\mathcal{F}([\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{0}]))), \end{aligned} \tag{4.11}$$

où

$$\beta(S(\mathbf{c}')) = \sum_{\mathbf{o}} P(\mathbf{o}) P(\mathbf{y}_o|\mathbf{o}) P(\mathbf{y}_c|\mathbf{c}' \oplus \mathcal{F}([\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{o}])), \tag{4.12}$$

pour tout $\mathbf{c}' \in GF(2)^{\ell(\mathbf{c})}$. Dans (4.12), $\beta(S(\mathbf{c}'))$ représente la somme des probabilités associées à toutes les combinaisons de $[\mathbf{o}, \mathbf{c} = \mathbf{c}' \oplus \mathcal{F}(\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{o})]$ qui partent de l'état $S(\mathbf{c}')$ dans le treillis sur $\mathbf{o}$.

A présent, nous allons démontrer que $\beta(S(\mathbf{c}'))$ dans (4.12) peut être calculé de manière récursive sur les $\ell(\mathbf{o})$ bits de $\mathbf{o}$, pour tout $\mathbf{c}' \in GF(2)^{\ell(\mathbf{c})}$. Pour cela, nous considérons que $\bar{\mathbf{o}}^j = [0 \dots 0, o_{j+1} \dots o_{\ell(\mathbf{o})}]$ et $\mathbf{y}_{\bar{o}}^j = [0 \dots 0, y_{o_{j+1}} \dots y_{o_{\ell(\mathbf{o})}}]$. De plus, nous définissons

$$\beta_j(S(\mathbf{c}')) = \sum_{\bar{\mathbf{o}}^j} P(\bar{\mathbf{o}}^j) P(\mathbf{y}_{\bar{o}}^j|\bar{\mathbf{o}}^j) P(\mathbf{y}_c|\mathbf{c}' \oplus \mathcal{F}([\mathbf{0}, \mathbf{0}, \mathbf{0}, \bar{\mathbf{o}}^j])), \tag{4.13}$$

comme la probabilité associée à l'état $S(\mathbf{c}') \in \{0 \dots 2^{\ell(\mathbf{c})} - 1\}$ à l'instant $j \in \{0 \dots \ell(\mathbf{o})\}$ dans le treillis sur $\mathbf{o}$.

En appliquant (4.13) à l'état $S(\mathbf{c}')$ à l'instant $j-1$, nous obtenons

$$\begin{aligned} \beta_{j-1}(S(\mathbf{c}')) &= \sum_{\bar{\mathbf{o}}^{j-1}} P(\bar{\mathbf{o}}^{j-1}) P(\mathbf{y}_{\bar{o}}^{j-1}|\bar{\mathbf{o}}^{j-1}) P(\mathbf{y}_c|\mathbf{c}' \oplus \mathcal{F}([\mathbf{0}, \mathbf{0}, \mathbf{0}, \bar{\mathbf{o}}^{j-1}])) \\ &= P(o_j = 0) P(y_{o_j}|o_j = 0) \sum_{\bar{\mathbf{o}}^j} P(\bar{\mathbf{o}}^j) P(\mathbf{y}_{\bar{o}}^j|\bar{\mathbf{o}}^j) P(\mathbf{y}_c|\mathbf{c}' \oplus \mathcal{F}([\mathbf{0}, \mathbf{0}, \mathbf{0}, \bar{\mathbf{o}}^j])) \\ &+ P(o_j = 1) P(y_{o_j}|o_j = 1) \sum_{\bar{\mathbf{o}}^j} P(\bar{\mathbf{o}}^j) P(\mathbf{y}_{\bar{o}}^j|\bar{\mathbf{o}}^j) P(\mathbf{y}_c|\mathbf{c}' \oplus \boldsymbol{\pi}(o_j) \oplus \mathcal{F}([\mathbf{0}, \mathbf{0}, \mathbf{0}, \bar{\mathbf{o}}^j])) \\ &= P(o_j = 0) P(y_{o_j}|o_j = 0) \cdot \beta_j(S(\mathbf{c}')) \\ &+ P(o_j = 1) P(y_{o_j}|o_j = 1) \cdot \beta_j(S(\mathbf{c}' \oplus \boldsymbol{\pi}(o_j))). \end{aligned} \tag{4.14}$$

A l'initialisation, c'est-à-dire quand $j = \ell(\mathbf{o})$,

$$\beta_{\ell(\mathbf{o})}(S(\mathbf{c}')) = P(\mathbf{y}_c|\mathbf{c}'), \text{ pour tout } \mathbf{c}' \in GF(2)^{\ell(\mathbf{c})}. \tag{4.15}$$

L'équation finale dans (4.14) est la clef pour calculer $\beta(S(\mathbf{c}'))$ par une méthode récursive (dans le sens indirect) sur les bits de $\mathbf{o}$. Après $\ell(\mathbf{o})$ itérations, $\beta_0(S(\mathbf{c}')) = \beta(S(\mathbf{c}'))$ dans (4.12), pour tout $\mathbf{c}' \in GF(2)^{\ell(\mathbf{c})}$. Par ailleurs, en étudiant l'équation récursive présentée dans (4.14), nous pouvons constater que les coefficients $\beta_j(S(\mathbf{c}'))$ correspondent exactement aux coefficients intervenant lors de la phase dans le sens indirect de l'algorithme BCJR [65]. Les coefficients $\beta(S(\mathbf{c}'))$ peuvent donc être calculés en effectuant une étape partielle dans le sens indirect de l'algorithme BCJR (en commençant par le dernier bit de $\mathbf{o}$ et en terminant par le premier bit de $\mathbf{o}$). Les étapes pour déterminer l'estimateur optimal $\widehat{\mathbf{u}}$ dans (4.8) avec la méthode présentée sont résumées ci-dessous :

Etape 1 : Initialiser $\beta_{\ell(\mathbf{o})}(S(\mathbf{c}'))$ en respectant (4.15).
Etape 2 : Déterminer $\beta_j(S(\mathbf{c}'))$, pour tout $\mathbf{c}' \in GF(2)^{\ell(\mathbf{c})}$ et pour tout $j = \ell(\mathbf{r}) - 1 \dots 0$, en utilisant (4.14) dans le sens indirect.
Etape 3 : Pour chaque $\mathbf{u} \in \Omega_u$, calculer la métrique $\mathcal{M}(\mathbf{u}) = P(\mathbf{y}_u|\mathbf{u})\beta_0(S(\mathcal{F}([\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{0}])))$ en considérant que $\mathbf{k}$ et $\mathbf{p}$ sont connus.
Etape 4 : L'estimateur optimal $\widehat{\mathbf{u}}$ correspond à l'élément $\mathbf{u} \in \Omega_u$ qui maximise $\mathcal{M}(\mathbf{u})$.

Avec cette méthode, nous pouvons déterminer l'estimateur optimal $\widehat{\mathbf{u}}$ avec une complexité $\mathcal{O}(\ell(\mathbf{o})2^{\ell(\mathbf{c})})$, comparé à $\mathcal{O}(|\Omega_u|2^{\ell(\mathbf{o})})$ avec la méthode basique (sans treillis).

Calcul approché de Ψ

En pratique, la plupart des CRCs ont une longueur supérieure à 16 bits et l'ordre de complexité obtenu avec la méthode précédente est encore trop important pour fournir une solution satisfaisante. Un calcul approché (basé sur la méthode proposée dans le partie 3.5.2) consiste à découper le CRC en M_p partitions de $\ell(\mathbf{c})/M_p$ bits et de considérer ces partitions indépendantes les unes des autres. Dans ce cas, $\mathbf{y}_c = [\mathbf{y}_{c_1} \dots \mathbf{y}_{c_{M_p}}]$. En utilisant cette hypothèse d'indépendance, Ψ dans (4.10) devient

$$\Psi \approx \Psi_1 \cdot \Psi_2 \cdots \Psi_{M_p} = \prod_{m=1}^{M_p} \Psi_m, \tag{4.16}$$

avec

$$\Psi_m = \sum_{\mathbf{o}} P(\mathbf{o})P(\mathbf{y}_o|\mathbf{o})P(\mathbf{y}_{c_m}|\mathcal{F}_m([\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{o}])), \tag{4.17}$$

dans laquelle $\mathcal{F}_m$ représente la fonction d'encodage associée à la m-ième partition du CRC, c'est-à-dire basée sur les colonnes allant de $(m-1) \cdot \frac{\ell(\mathbf{c})}{M_p} + 1$ à $m \cdot \frac{\ell(\mathbf{c})}{M_p}$ dans la matrice $\mathbf{\Pi}$. Chaque Ψ_m est déterminé en utilisant la méthode exacte présentée précédemment. Dans ce cas, chaque treillis est composé de seulement $2^{\ell(\mathbf{c})/M_p}$ états (alors que le treillis possédait $2^{\ell(\mathbf{c})}$ états pour le calcul de Ψ). Par conséquent, la métrique $\mathcal{M}(\mathbf{u})$ associée à une valeur de $\mathbf{u} \in \Omega_u$ est définie par

$$\mathcal{M}(\mathbf{u}) = P(\mathbf{y}_u|\mathbf{u}) \prod_{m=1}^{M_p} \beta^m(S(\mathcal{F}_m([\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{0}]))), \tag{4.18}$$

où $\beta^m(S(\mathbf{c}'_m)) = \beta_0^m(S(\mathbf{c}'_m))$ et correspond à la probabilité associée à l'état $S(\mathbf{c}'_m)$ à l'instant $j = 0$ dans le m-ième treillis sur $\mathbf{o}$.

Exemple 2 *Le code systématique* $(11, 7)$*, dont le polynôme générateur* $g(z) = z^4 + z + 1$*, possède une matrice de parité*

$$\mathbf{\Pi} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Dans cet exemple, nous considérons que $\mathbf{k} = [0]$ *et* $\mathbf{p} = [1]$*. Nous cherchons à déterminer la valeur approchée du terme* Ψ *pour la valeur de* $\mathbf{u} = [1]$*. Par conséquent,* $\ell(\mathbf{k}) + \ell(\mathbf{p}) + \ell(\mathbf{u}) = 3$ *bits et* $\ell(\mathbf{o}) = 3$ *bits. Pour réduire la complexité du traitement, nous considérons que le CRC est découpé en* 2 *partitions de* 2 *bits. La matrice de parité* $\mathbf{\Pi}$ *peut donc être décomposée en deux sous-matrices* $[\mathbf{\Pi}_1, \mathbf{\Pi}_2]$ *tel que*

$$\mathbf{\Pi}_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad et \quad \mathbf{\Pi}_2 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

En utilisant cette représentation, nous pouvons construire deux treillis dans le sens indirect : le premier dépend de $\mathbf{\Pi}_1$ *et le deuxième de* $\mathbf{\Pi}_2$*. Ces deux treillis sont exposés sur la figure 4.3. Ces treillis prennent en compte les bits de* $\mathbf{o}$ *et impliquent donc les* 4 *dernières lignes de chaque sous-matrice de parité. La valeur de* Ψ *quand* $[\mathbf{k}, \mathbf{p}, \mathbf{u}] = [0, 1, 1]$ *peut ensuite être évaluée en multipliant* $\Psi_1 = \beta^1(S(\mathcal{F}_1([\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{0}])) = 3)$ *dans le premier treillis par* $\Psi_2 = \beta^2(S(\mathcal{F}_2([\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{0}])) = 3)$ *dans le deuxième treillis. Ce principe peut être facilement étendu à un nombre de partitions supérieur à* 2*.*

Avec cette méthode approchée, la complexité totale de l'estimation (reposant sur la détermination de $\widehat{\mathbf{u}}$) devient $\mathcal{O}(\ell(\mathbf{o}) M_p 2^{\ell(\mathbf{c})/M_p})$, au prix d'une légère sous-optimalité.

4.5 Application à la norme 802.11

Dans cette partie, nous proposons un modèle de couche perméable approprié aux couches PHY et MAC du terminal WiFi. Ce dispositif exploite les concepts qui ont été expliqués dans les parties précédentes. Le format des paquets PHY et MAC est brièvement rappelé dans les

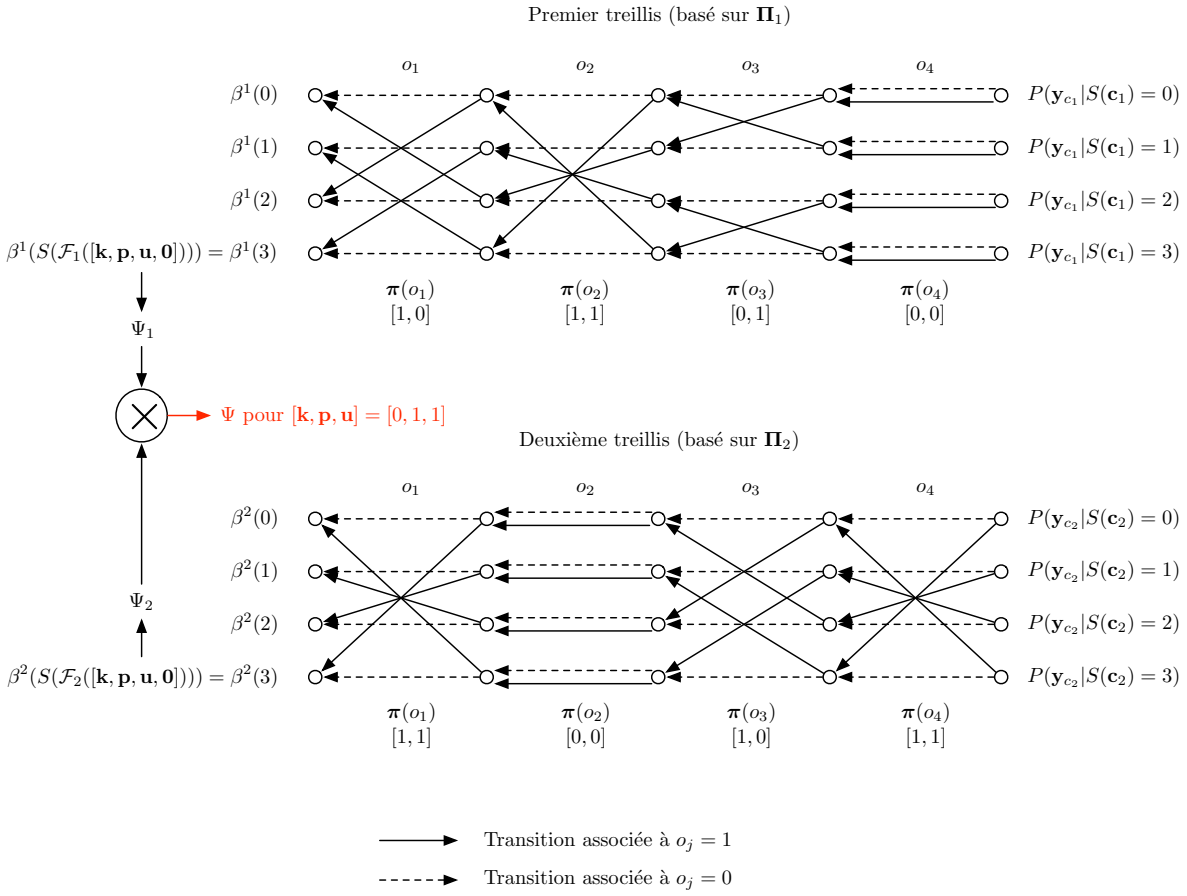


FIG. 4.3 – Illustration du calcul approché de Ψ à partir des treillis

parties 4.5.1 et 4.5.2. Les redondances intra-couche et inter-couches sont identifiées dans la partie 4.5.3. Le détail des traitements ainsi qu'un schéma général sont finalement présentés dans les parties 4.5.4, 4.5.5 et 4.5.6.

4.5.1 Description de la couche PHY

La couche Physique de la norme 802.11 a été largement présentée dans la partie 2.3.1. Dans ce paragraphe, nous rappelons uniquement les informations qui nous sont utiles.

La norme 802.11 fournit des débits de transmission de 1 ou 2 Mbps dans la bande des 2.4 GHz. Deux techniques peuvent être utilisées : le FHSS ou le DSSS. Avec la méthode DSSS, un code de Barker de 11 *chips* est employé pour étaler le flux de 1 Mbps. Cette opération génère un signal en bande de base de 11 MHz. Une modulation BPSK ou QPSK est ensuite appliquée pour fournir les débits de 1 ou 2 Mbps respectivement.

Le format des paquets PHY codés en DSSS est rappelé sur la figure 4.5. Le préambule et

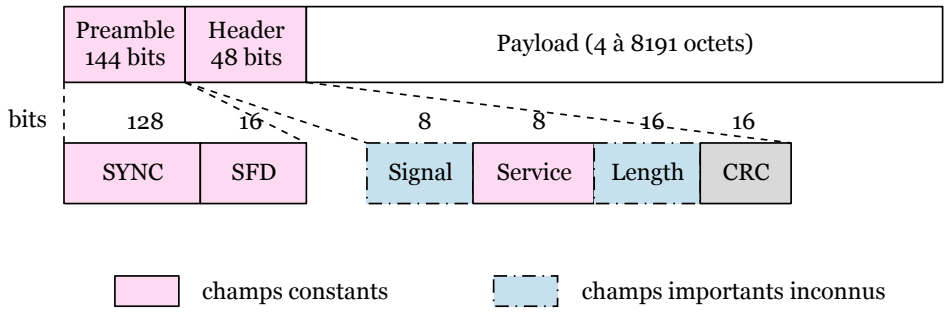


FIG. 4.4 – Format des paquets PHY et identification des champs

l'en-tête des paquets sont toujours transmis en utilisant la modulation DBPSK. En revanche, le schéma de modulation appliqué aux données contenues dans la *payload* est variable. Il dépend de la qualité du canal. Lorsque le canal est perturbé, la modulation la plus robuste est employée (BPSK). A l'inverse, la modulation QPSK est utilisée pour fournir des débits plus élevés. Les champs *SYNC* et *SFD* représentent une séquence d'apprentissage de 144 bits. Cette séquence permet au récepteur de se synchroniser sur le flux. Ces deux champs ne sont pas protégés par le CRC. Dans cette étude, nous proposons de les utiliser également pour estimer la variance du bruit introduit par le canal (voir la partie 4.5.6).

Le CRC de 2 octets contenu dans l'en-tête protège les champs *Signal*, *Service* et *Length*. Sa fonction d'encodage est notée $\mathcal{F}^{\mathrm{PHY}}$. La *payload*, contenant un fragment MAC, n'est pas protégée dans cette couche. Le champ *Service* est réservé pour des recommandations futures et sa valeur est fixée à 00_{16}. Il est inclus dans le vecteur $\mathbf{k}^{\mathrm{PHY}}$, en accord avec les notations introduites dans la partie 4.3. Le champ *Signal* spécifie la modulation de la *payload*. Il est égal à $0A_{16}$ pour la modulation BPSK et à 14_{16} pour la modulation QPSK. Le champ *Length* définit le nombre de microsecondes requis pour transmettre la *payload*. Sa valeur dépend à la fois du débit et de la taille de la *payload*. *Signal* et *Length* caractérisent donc des paramètres importants inconnus couverts par le CRC et sont insérés dans le vecteur $\mathbf{u}^{\mathrm{PHY}}$. Dans cette couche, $\mathbf{p}^{\mathrm{PHY}} = \mathbf{o}^{\mathrm{PHY}} = \emptyset$.

4.5.2 Description de la couche MAC

Une description exhaustive de la couche Liaison 802.11 a été fournie dans la partie 2.3.2. Dans ce paragraphe, nous mentionnons juste les points essentiels qui nous sont utiles.

Le format des fragments MAC est illustré sur la figure 4.5. Dans ces paquets, le CRC de 4 octets protège à la fois les champs de l'en-tête ainsi que les informations contenues dans la *payload*. Sa fonction d'encodage est définie par $\mathcal{F}^{\mathrm{MAC}}$.

Dans la suite, nous considérons une transmission de données vidéo entre un serveur et un terminal (sens descendant). Au niveau du point d'accès WiFi, les retransmissions ne sont

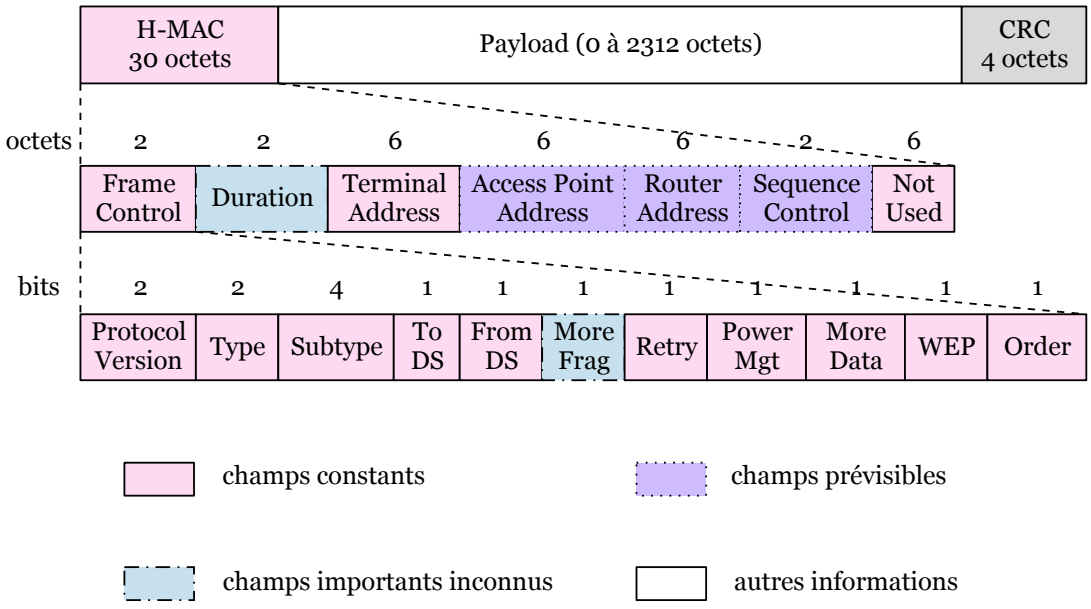


FIG. 4.5 – Format des fragments MAC et classification des champs

pas autorisées et le mode économie d'énergie est désactivé. Par ailleurs, les fragments MAC ne sont pas cryptés et sont transmis dans l'ordre. Sous ces hypothèses, les 2 octets du champ *Frame Control*, excepté le bit *More Frag*, sont supposés connus. Le champ *Terminal Address* est également considéré connu du récepteur. De plus, le dernier champ de l'en-tête MAC n'est pas utilisé dans ces conditions et ses bits sont tous fixés à 0. En reprenant les notations de la partie 4.3, tous ces champs sont inclus dans le vecteur $\mathbf{k}^{\mathrm{MAC}}$.

Le champ *Access Point Address* a été transmis au terminal durant la procédure de réservation du canal (RTS-CTS). Cette adresse peut donc être prédite par le récepteur. D'autre part, le point d'accès est souvent connecté à un seul routeur. Dans ces conditions, le champ *Router Address* peut être extrait des autres paquets d'information en provenance d'Internet. Le champ *Sequence Control* contient deux informations : un numéro de séquence et un numéro de fragment. La transmission des fragments étant ordonnée, ces paramètres peuvent facilement être déterminés : le numéro de séquence est incrémenté de 1 lors de l'échange des paquets RTS-CTS et le numéro de fragment est incrémenté de 1 à chaque fragment reçu. Tous ces champs déductibles sont incorporés dans le vecteur $\mathbf{p}^{\mathrm{MAC}}$.

Le bit *More Frag* spécifie si le fragment courant est le dernier fragment d'une séquence. Le champ *Duration* indique la durée de transmission estimée (en microsecondes) du prochain fragment et des informations de contrôle. Sa valeur dépend de la modulation du fragment courant et de la taille du futur fragment. Ces deux champs ne peuvent pas être prédits totalement par le récepteur. Il sont insérés dans le vecteur $\mathbf{u}^{\mathrm{MAC}}$.

Finalement, la *payload* contient les données utiles à transmettre. Ses données ne sont pas utilisées par la couche MAC mais le CRC les protège. Elles sont donc introduites dans le

vecteur $\mathbf{o}^{\mathrm{MAC}}$.

4.5.3 Identification des redondances

Les corrélations intra-couche et inter-couches facilitent la récupération des en-têtes bruités puisqu'elles permettent de prédire le vecteur $\mathbf{p}$ et de construire l'ensemble Ω_u défini dans la partie 4.3. La mise en évidence de ces corrélations repose principalement sur les spécifications de la couche MAC. Ces dernières ont été largement expliquées dans la partie 2.3.2. Pour faciliter la compréhension, le principe général est résumé dans ce chapitre.

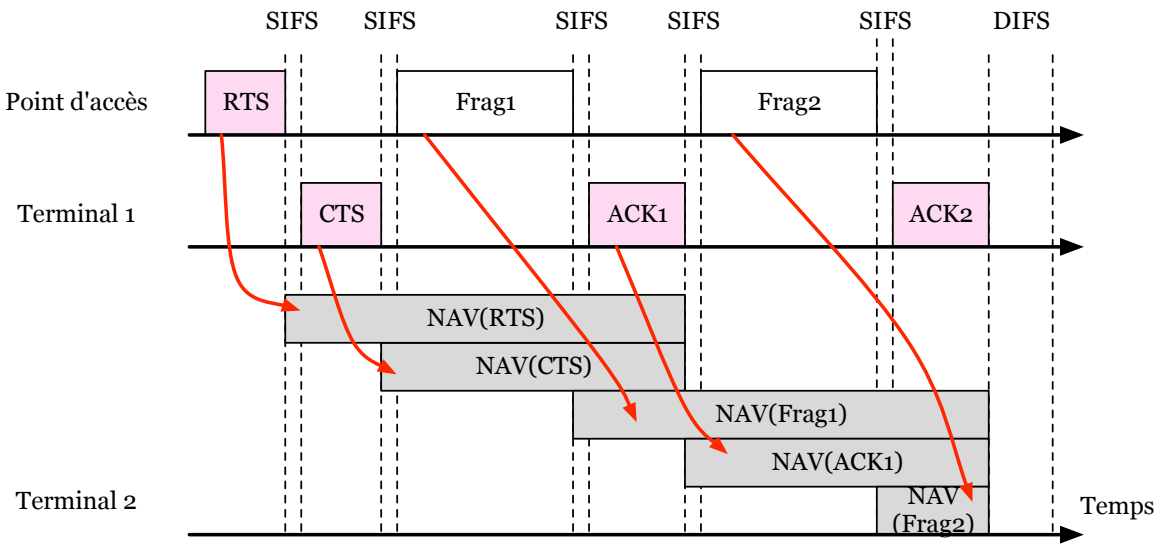


FIG. 4.6 – Protocole de transmission de la couche MAC

La figure 4.6 illustre le protocole de transmission de la couche MAC. Dans cet exemple, le paquet IP est réparti en deux fragments. La transmission est initialisée par une procédure de réservation du support physique consistant en un échange de paquets RTS et CTS entre le point d'accès et le terminal. Les fragments sont ensuite transmis au récepteur qui les acquitte (ACK). Dans cette étude, nous considérons que les paquets RTS, CTS et MAC sont correctement reçus. Seules les erreurs intervenant dans les fragments sont traitées. Un intervalle SIFS sépare chaque paquet pour éviter les collisions. Quand tous les fragments d'un paquet IP ont été transmis au terminal, un intervalle DIFS précède la prochaine procédure de réservation du canal. Un champ *Duration* est inclus dans chaque paquet MAC et sa valeur indique le nombre de microsecondes nécessaires pour transmettre le prochain fragment. Elle permet d'ajuster le NAV des autres terminaux (cette propriété est représentée à l'aide des flèches rouges sur la figure 4.6). Les autres stations ne peuvent pas communiquer pendant la période définie par le NAV.

Supposons que D_n^{MAC} et B_n^{PHY} représentent la valeur de *Duration* et le débit de transmission (codé par *Signal*) associés au n-ième paquet transmis par le point d'accès (correspondant

soit à un RTS, soit à un fragment). En suivant les procédures du standard 802.11, nous pouvons écrire que D_n^{MAC} est défini par

$$D_n^{\mathrm{MAC}} = 3T_{\mathrm{SIFS}} + 3T_{\mathrm{OVH}} + 2\frac{\ell_{\text{C-A}}}{B_n^{\mathrm{PHY}}} + \frac{\ell(\mathbf{x}_{n+1}^{\mathrm{PHY}})}{B_n^{\mathrm{PHY}}}, \tag{4.19}$$

sauf dans le dernier fragment d'une séquence, c'est-à-dire quand la valeur de *More Frag* $M_n^{\mathrm{MAC}} = 0$. Dans ce cas, on a

$$D_n^{\mathrm{MAC}} = T_{\mathrm{SIFS}} + T_{\mathrm{OVH}} + \frac{\ell_{\text{C-A}}}{B_n^{\mathrm{PHY}}}. \tag{4.20}$$

Dans (4.19) et (4.20), T_{SIFS} définit la durée d'un intervalle SIFS et T_{OVH} représente le temps nécessaire pour transmettre le préambule et l'en-tête des paquets PHY à 1 Mbps. Les autres termes de (4.19) dépendent du débit courant B_n^{PHY}. Les paquets CTS et ACK ont la même taille constante de $\ell_{\text{C-A}}$ bits et $\ell_{\text{C-A}}/B_n^{\mathrm{PHY}}$ correspond donc à la durée requise pour transmettre ces paquets. Finalement, $\ell(\mathbf{x}_{n+1}^{\mathrm{PHY}})/B_n^{\mathrm{PHY}}$ fait référence à la durée de transmission de la prochaine *payload* PHY de $\ell(\mathbf{x}_{n+1}^{\mathrm{PHY}})$ bits.

4.5.4 Méthode de récupération des en-têtes PHY

Dans le n-ième paquet arrivant à la couche PHY, les observations associées à $\mathbf{k}_n^{\mathrm{PHY}}$, $\mathbf{u}_n^{\mathrm{PHY}}$ et $\mathbf{c}_n^{\mathrm{PHY}}$ définis dans la partie 4.5.1 sont rassemblées dans $\mathbf{y}_n^{\mathrm{PHY}} = [\mathbf{y}_{k,n}^{\mathrm{PHY}}, \mathbf{y}_{u,n}^{\mathrm{PHY}}, \mathbf{y}_{c,n}^{\mathrm{PHY}}]$. Par ailleurs, $\mathbf{y}_{x,n}^{\mathrm{PHY}}$ représente les observations attachées aux $\ell(\mathbf{x}_n^{\mathrm{PHY}})$ bits de la *payload*.

Le nombre de combinaisons possibles pour $\mathbf{u}^{\mathrm{PHY}}$ peut être significativement réduit en exploitant le champ *Duration* contenu dans le paquet MAC précédemment reçu. Plus précisément, en utilisant B_{n-1}^{PHY} et D_{n-1}^{MAC}, nous pouvons facilement déduire $\ell(\mathbf{x}_n^{\mathrm{PHY}})$ de (4.19) en appliquant

$$\ell(\mathbf{x}_n^{\mathrm{PHY}}) = \left(D_{n-1}^{\mathrm{MAC}} - 3T_{\mathrm{SIFS}} - 3T_{\mathrm{OVH}} - 2\frac{\ell_{\text{C-A}}}{B_{n-1}^{\mathrm{PHY}}} \right) B_{n-1}^{\mathrm{PHY}}. \tag{4.21}$$

La durée L_n^{PHY}, codée dans le champ *Length* du paquet PHY courant, peut être ensuite évaluée à partir de B_n^{PHY} par

$$L_n^{\mathrm{PHY}} = \frac{\ell(\mathbf{x}_n^{\mathrm{PHY}})}{B_n^{\mathrm{PHY}}}. \tag{4.22}$$

Dans (4.21), la valeur de $\ell(\mathbf{x}_n^{\mathrm{PHY}})$ est totalement déterminée si les en-têtes des paquets PHY et MAC précédents ont été correctement reconstruits. D'autre part, en s'appuyant sur (4.22), nous pouvons constater que L_n^{PHY} est limité à seulement deux combinaisons qui sont définies par la valeur de B_n^{PHY}. Ces possibilités sont stockées dans l'ensemble $\Omega_{u,n}^{\mathrm{PHY}}$.

En intégrant ces propriétés structurelles dans (4.9), nous obtenons l'estimateur des en-têtes de la couche PHY

$$\widehat{\mathbf{u}}_n^{\mathrm{PHY}} = \arg \max_{\mathbf{u}_n^{\mathrm{PHY}} \in \Omega_{u,n}^{\mathrm{PHY}}} P(\mathbf{y}_{u,n}^{\mathrm{PHY}} | \mathbf{u}_n^{\mathrm{PHY}}) P(\mathbf{y}_{c,n}^{\mathrm{PHY}} | \mathbf{c}_n^{\mathrm{PHY}}), \tag{4.23}$$

où $\mathbf{c}_n^{\mathrm{PHY}} = \mathcal{F}^{\mathrm{PHY}}\left(\left[\mathbf{k}_n^{\mathrm{PHY}}, \mathbf{u}_n^{\mathrm{PHY}}\right]\right)$.

4.5.5 Méthode de récupération des en-têtes MAC

Les informations souples contenues dans $\mathbf{y}_{x,n}^{\mathrm{PHY}}$ arrivent ensuite à l'entrée de la couche MAC. $\mathbf{y}_{x,n}^{\mathrm{PHY}}$ contient les observations associées aux vecteurs $\mathbf{k}_n^{\mathrm{MAC}}$, $\mathbf{p}_n^{\mathrm{MAC}}$, $\mathbf{u}_n^{\mathrm{MAC}}$, $\mathbf{o}_n^{\mathrm{MAC}}$ et $\mathbf{c}_n^{\mathrm{MAC}}$ spécifiés dans la partie 4.5.2 tel que $\mathbf{y}_n^{\mathrm{MAC}} = \mathbf{y}_{x,n}^{\mathrm{PHY}} = [\mathbf{y}_{k,n}^{\mathrm{MAC}}, \mathbf{y}_{p,n}^{\mathrm{MAC}}, \mathbf{y}_{u,n}^{\mathrm{MAC}}, \mathbf{y}_{o,n}^{\mathrm{MAC}}, \mathbf{y}_{c,n}^{\mathrm{MAC}}]$.

Le nombre de combinaisons possibles pour $\mathbf{u}_n^{\mathrm{MAC}}$ peut être considérablement réduit en exploitant les propriétés définies dans (4.19) and (4.20). En effet, nous pouvons noter que D_n^{MAC} est totalement déterminé quand $M_n^{\mathrm{MAC}} = 0$. Quand $M_n^{\mathrm{MAC}} = 1$, la valeur de *Duration* dépend de la taille de la prochaine *payload* PHY. En considérant que le futur fragment MAC contient un nombre entier d'octets, les valeurs possibles de $\ell(\mathbf{x}_{n+1}^{\mathrm{PHY}})$ dans (4.19) sont données par

$$\ell(\mathbf{x}_{n+1}^{\mathrm{PHY}}) = \ell_{\mathrm{HDR}} + 8 \cdot i, \tag{4.24}$$

dans laquelle $i = 1, 2 \ldots 2312$. Dans (4.24), ℓ_{HDR} spécifie la taille connue de l'en-tête des fragments MAC. En combinant les corrélations introduites par (4.19), (4.20) et (4.24), on peut montrer que $\mathbf{u}_n^{\mathrm{MAC}}$ est limité à 2313 combinaisons qui sont insérées dans l'ensemble $\Omega_{u,n}^{\mathrm{MAC}}$.

En intégrant ces propriétés dans (4.8), on obtient l'estimateur des en-têtes de la couche MAC

$$\widehat{\mathbf{u}}_n^{\mathrm{MAC}} = \arg \max_{\mathbf{u}_n^{\mathrm{MAC}} \in \Omega_{u,n}^{\mathrm{MAC}}} P(\mathbf{y}_{u,n}^{\mathrm{MAC}} | \mathbf{u}_n^{\mathrm{MAC}}) \Psi(\mathbf{k}_n^{\mathrm{MAC}}, \mathbf{p}_n^{\mathrm{MAC}}, \mathbf{u}_n^{\mathrm{MAC}}, \mathbf{y}_{o,n}^{\mathrm{MAC}}, \mathbf{y}_{c,n}^{\mathrm{MAC}}), \tag{4.25}$$

où le second terme peut être calculé avec les méthodes optimisées introduites dans la partie 4.4.2.

4.5.6 Schéma général

La figure 4.7 illustre les mécanismes robustes de récupération des en-têtes adaptés aux couches PHY et MAC du terminal WiFi. Cette figure met en évidence les échanges d'information à travers les couches et entre les paquets consécutifs. Ces corrélations ont été introduites dans les parties 4.5.4 et 4.5.5.

D'autre part, nous considérons que $\mathbf{y}_{s,n}^{\mathrm{PHY}}$ représente les observations de la séquence d'apprentissage $\mathbf{s}^{\mathrm{PHY}}$ (préambule des paquets PHY) de $\ell(\mathbf{s}^{\mathrm{PHY}})$ bits. Comme nous l'avons déjà précisé dans la partie 4.5.1, $\mathbf{s}^{\mathrm{PHY}}$ permet au récepteur de se synchroniser sur le flux. Dans cette étude, nous proposons d'estimer simultanément la variance du bruit σ^2 à partir de $\mathbf{s}^{\mathrm{PHY}}$ et de $\mathbf{y}_{s,n}^{\mathrm{PHY}}$. Cette mesure est essentielle pour travailler avec les informations souples puisqu'elle permet de déterminer les vraisemblances. L'estimateur $\widehat{\sigma}^2$ de la puissance du bruit est donné par

$$\widehat{\sigma}^2 = \frac{1}{\ell(\mathbf{s}^{\mathrm{PHY}})} \|\mathbf{y}_{s,n}^{\mathrm{PHY}} - \mathbf{s}^{\mathrm{PHY}}\|^2, \tag{4.26}$$

où $\|\cdot\|$ représente la distance euclidienne.

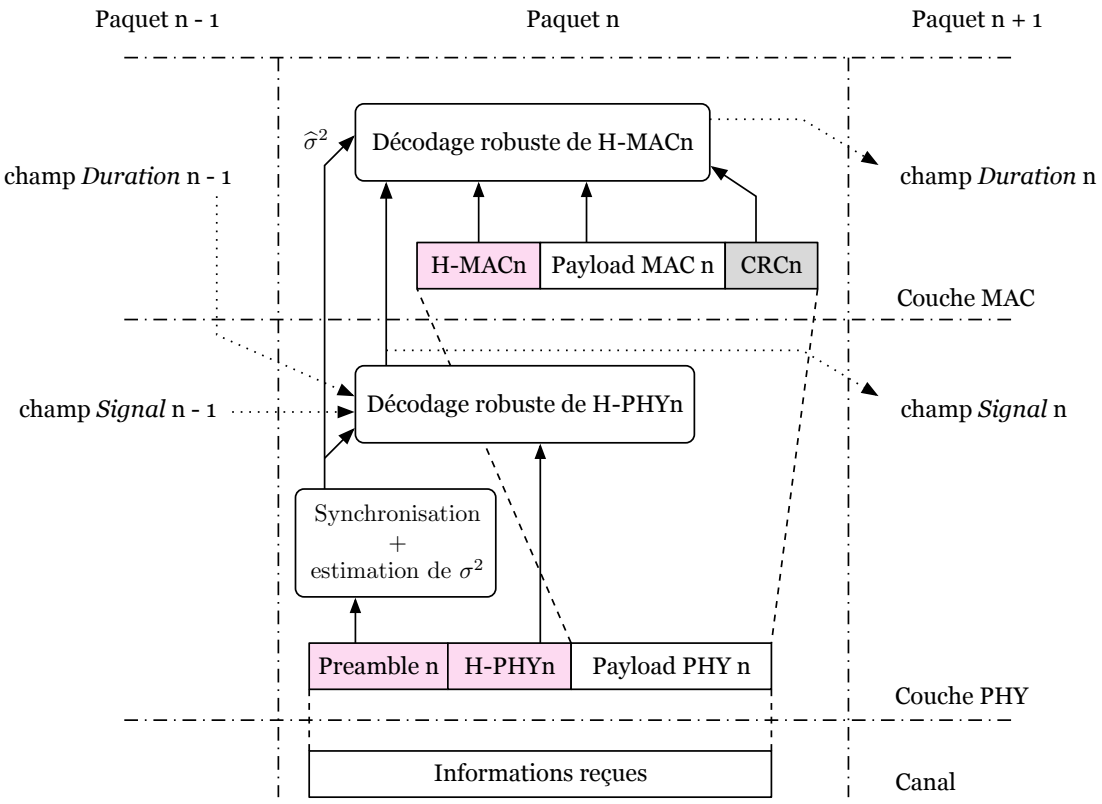


FIG. 4.7 – Schéma robuste de décodage des en-têtes pour les couches PHY et MAC

La complexité algorithmique peut être minimisée en désactivant le traitement robuste quand aucune erreur n'est détectée par le CRC. Dans ce cas, les informations contenues dans l'en-tête des paquets peuvent être utilisées directement par une procédure standard. Le décodage robuste peut également être désactivé quand la qualité des informations souples est trop pauvre, c'est-à-dire lorsque la puissance du signal reçu est inférieure à un seuil pré-défini. Dans cette situation, les fragments peuvent être retransmis par le point d'accès. Ce principe permet de réactiver les retransmissions sans se baser sur le calcul traditionnel du CRC.

Remarque 4 *Le champ* Service *peut être utilisé pour séparer les paquets reçus à la couche PHY. Par exemple, ce champ peut être fixé à* 00_{16} *dans les paquets qui doivent suivre une procédure standard et à* FF_{16} *dans les paquets qui peuvent être traités par des couches incorporant des algorithmes robustes. Cette méthode permet d'orienter les paquets vers une pile protocolaire adaptée à leur contenu au niveau le plus bas.*

4.6 Résultats de simulation

Le schéma perméable de la figure 4.7 a été implémenté. Nous avons étudié chaque couche séparément en simulant un système de transmission composé d'un point d'accès, d'un canal et d'un terminal.

Le point d'accès crée les paquets PHY et MAC en respectant les protocoles définis dans la partie 4.5. La *payload* MAC contient une quantité variable de bits générés aléatoirement. L'émetteur module les données en BPSK dans tous les paquets transmis et transmet les informations modulées sur un canal AWGN. Dans ces simulations, le codage/décodage canal du protocole DSSS de la norme 802.11 a été désactivé : le SNR est donc déterminé par rapport à l'énergie binaire (il n'est pas basé sur les *chips*). Au niveau du récepteur, trois types de méthodes de récupération des en-têtes sont considérés :

1. Un décodeur *standard* qui réalise une décision dure sur les informations entrantes.
2. Un décodeur *robust* qui exploite à la fois les redondances structurelles et les informations souples, en négligeant cependant les propriétés du CRC.
3. Un décodeur *CRC-robust* combinant toutes les sources de redondances précédentes ainsi que les redondances apportées par le CRC.

L'analyse des performances est basée sur des courbes traçant l'évolution du EHR (*Erroneous Header Rate*) en fonction du SNR. Les résultats sont décrits ci-dessous.

Sur la figure 4.8, les trois types de décodeurs intégrés dans la couche PHY (*standard*, *robust* et *CRC-robust*) sont comparés. Pour étudier les performances des couches PHY et MAC séparément, nous considérons que le champ *Duration* contenu dans le paquet MAC précédent a été parfaitement reçu. Il est évident que le décodeur standard est dépassé par les deux décodeurs robustes. Nous pouvons constater que le décodeur *robust* atteint un EHR de 10^{-5} lorsque le SNR est supérieur à 4 dB. Ce même résultat est obtenu pour le décodeur *CRC-robust* pour un SNR d'environ 2 dB. Avec la méthode standard, un SNR de 15 dB est nécessaire pour atteindre un EHR comparable. A la couche PHY, des gains de codage significatifs sont donc observés pour les deux traitements robustes. Nous pouvons également noter que les performances obtenues sont principalement liées à l'exploitation des redondances intra-couche et inter-couches. Les propriétés du CRC ont un impact mineur sur les performances générales. Par ailleurs, la complexité algorithmique du traitement est relativement faible puisque l'estimateur 4.23 est utilisé pour réaliser le décodage.

La figure 4.9 compare les résultats obtenus par les trois types de décodeurs au niveau de la couche MAC. Pour analyser uniquement les performances associées à cette couche, nous supposons que le champ *Signal* (indiquant le débit de transmission) contenu dans le paquet PHY courant a été correctement décodé par le récepteur. Deux tailles de *payload* ont été considérées : 50 et 100 octets. Pour réduire la complexité du décodage robuste basé sur le CRC, la méthode sous-optimale présentée dans la partie 4.4.2 a été employée. Dans cette simulation, le CRC a été découpé en 4 blocs de 8 bits. Nous pouvons observer que

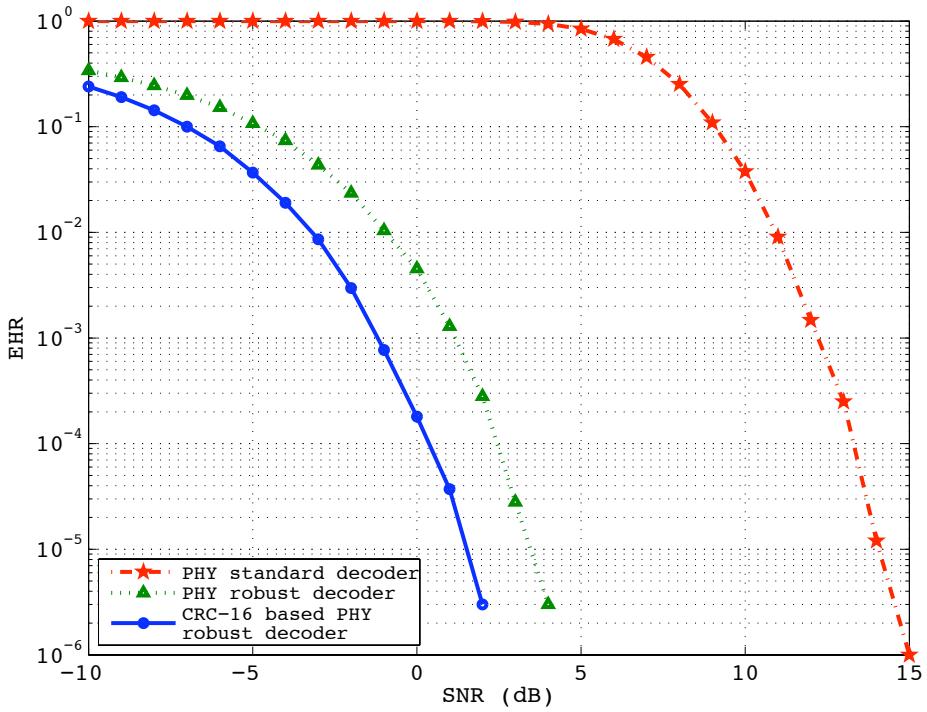


FIG. 4.8 – EHR en fonction du SNR pour la couche PHY

les courbes obtenues possèdent les mêmes caractéristiques que celles de la couche PHY. Les gains de codage sont cependant moins importants. Pour une *payload* de 100 octets, un EHR inférieur à 10^{-5} est atteint pour des valeurs de SNR de 11 dB, 14 dB et 15 dB en utilisant respectivement un décodeur *CRC-robust*, *robust* et *standard*.

Cependant, la méthode de décodage robuste basée sur l'exploitation du CRC possède une complexité accrue, à cause de la marginalisation dans (4.25). Plus la taille de la *payload* est élevée, plus le traitement est complexe. Pour réduire la complexité et améliorer les performances de la technique de récupération des en-têtes MAC, nous avons appliqué le principe du protocole UDP-Lite à la couche MAC. Nous avons introduit une couche MAC perméable, appelée MAC-Lite, dans laquelle le CRC ne protège que l'en-tête des fragments. Dans ce cas, $\mathbf{o}_n^{\text{MAC-L}} = \emptyset$ et (4.25) devient

$$\widehat{\mathbf{u}}_n^{\text{MAC-L}} = \arg \max_{\mathbf{u}_n^{\text{MAC-L}} \in \Omega_{u,n}^{\text{MAC-L}}} P(\mathbf{y}_{u,n}^{\text{MAC-L}} | \mathbf{u}_n^{\text{MAC-L}}) P(\mathbf{y}_{c,n}^{\text{MAC-L}} | \mathbf{c}_n^{\text{MAC-L}}), \tag{4.27}$$

où $\mathbf{c}_n^{\text{MAC-L}} = \mathcal{F}^{\text{MAC-L}}([\mathbf{k}_n^{\text{MAC-L}}, \mathbf{p}_n^{\text{MAC-L}}, \mathbf{u}_n^{\text{MAC-L}}])$ représente un simple calcul de CRC.

Les trois types de décodeurs appliqués à la couche MAC-Lite sont comparés sur la figure 4.10. Nous pouvons observer que les courbes associées aux décodages *robust* et *standard* des couches MAC et MAC-Lite sont identiques. Ce phénomène est normal puisque les informations apportées par le CRC ne sont pas utilisées par ces deux décodeurs. En revanche, le

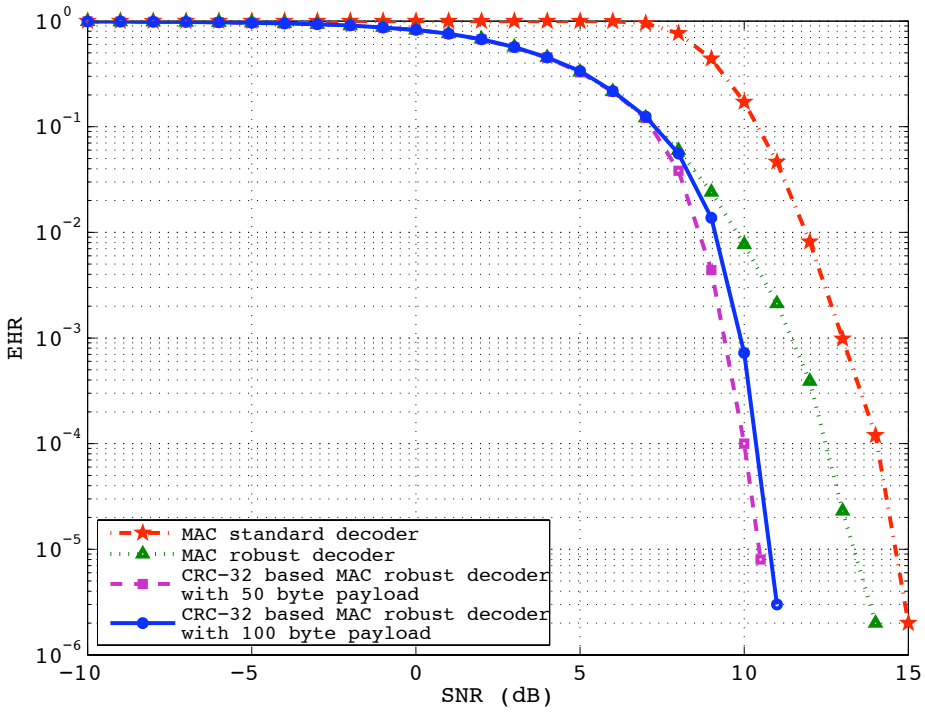


FIG. 4.9 – EHR en fonction du SNR pour la couche MAC

décodeur *CRC-robust* de la couche MAC-Lite est beaucoup plus efficace : un EHR inférieur à 10^{-5} est atteint quand le SNR dépasse 3 dB. D'autre part, la complexité associée au traitement robuste exploitant les propriétés du CRC est significativement réduite dans la couche MAC-Lite puisque l'estimateur (4.27) remplace (4.25).

En résumé, l'utilisation conjointe des protocoles robustes PHY et MAC-Lite dans le récepteur permet de corriger efficacement les en-têtes protocolaires des paquets PHY et MAC transmis. Avec ce mécanisme, pratiquement toutes les informations utiles des paquets sont transmises vers les couches supérieures à partir d'un SNR de 3 dB. Lorsque la couche perméable MAC est utilisée dans le terminal, les mêmes résultats sont obtenus pour un SNR supérieur à 11 dB et avec une complexité algorithmique beaucoup plus importante. Ces analyses démontrent donc l'intérêt de remplacer le protocole MAC standard par le protocole MAC-Lite pour aboutir à un modèle perméable efficace. Par ailleurs, il est important de noter que les gains de codage resteraient identiques si le codage/décodage canal DSSS avait été activé. Les courbes seraient juste décalées vers les faibles valeurs de SNR.

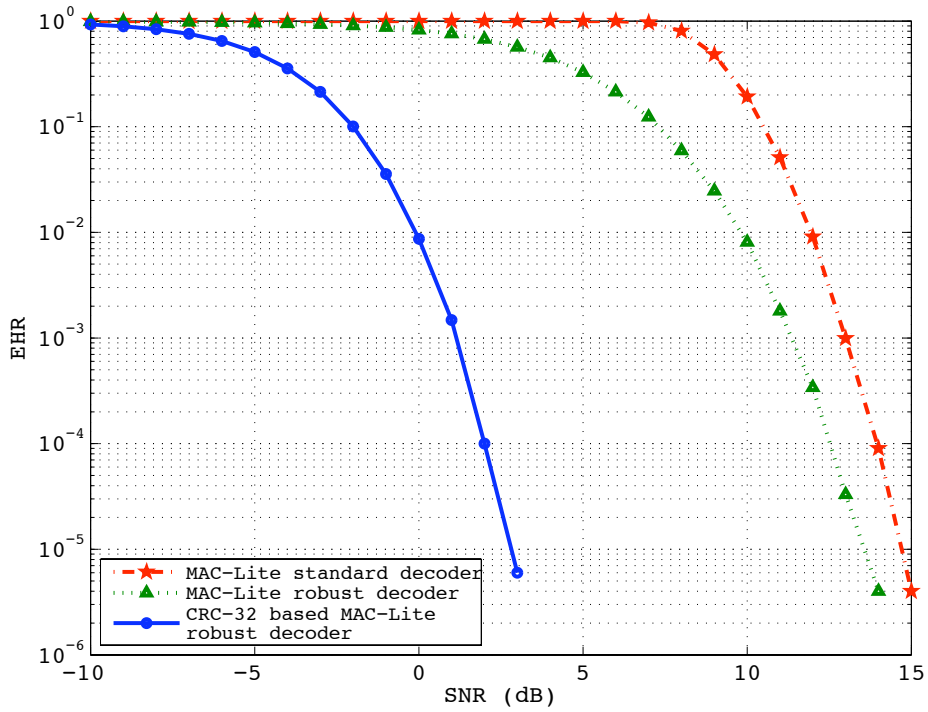


FIG. 4.10 – EHR en fonction du SNR pour la couche MAC-Lite

4.7 Conclusion

Dans ce chapitre, nous avons présenté un modèle de couche perméable qui est utile lors de la transmission robuste de vidéo. L'outil principal de cette solution consiste à corriger les champs d'information contenues dans l'en-tête des paquets. Il est basé sur un estimateur au sens MAP exploitant conjointement les propriétés structurelles de la pile protocolaire ainsi que les codes de détection d'erreurs contenus dans les paquets. Au niveau de chaque couche, les redondances intra-couche et inter-couches sont utilisées pour construire un ensemble réduit de combinaisons associées à l'en-tête. Le candidat le plus probable est ensuite obtenu en réalisant un décodage basé sur les informations du CRC (ou du checksum). Des simulations ont été réalisées sur les couches PHY et MAC de WiFi et les résultats obtenus sont concluants : les gains de codage peuvent atteindre 12 dB et la complexité algorithmique des solutions les plus efficaces est négligeable. Il faut noter que cette méthode peut facilement s'adapter à des schémas de transmission variés. Notre objectif futur est d'appliquer cette technique aux couches IP et UDP du récepteur (voir parties 2.3.3 et 2.3.4). Dans ce contexte, une alternative consisterait à combiner la technique proposée et le protocole ROHC pour renforcer les performances du traitement. Nous aboutirons alors à une pile protocolaire totalement perméable qui sera particulièrement bien adaptée aux techniques de décodage conjoint source-canal situées dans la couche APL.

Conclusion

Dans cette thèse, nous avons cherché à optimiser la transmission d'un flux vidéo entre un serveur et un terminal WiFi reliés à Internet. Nous avons montré que le maillon faible d'une telle transmission était associé au lien radio intervenant à l'extrémité du réseau (entre le point d'accès et le récepteur mobile). En effet, ce support fournit des débits limités et génère un taux d'erreurs important. Les points d'action principaux se situent donc au niveau du serveur, du point d'accès et du terminal. Dans cette thèse, nous avons développé deux axes d'étude complémentaires, destinés à améliorer la qualité du transport global :

1. un décodeur robuste exploitant les redondances inhérentes de la pile protocolaire,
2. un modèle de couche perméable basé sur la correction des en-têtes protocolaires.

Le décodeur robuste permet d'améliorer la qualité de la vidéo au niveau du récepteur. Cette méthode JSCD est basée sur un traitement séquentiel exploitant conjointement les propriétés sémantiques et syntaxiques du codeur source ainsi que le CRC contenu dans les fragments de la couche MAC 802.11. Cette technique a été appliquée pour décoder les résidus de prédiction générés par le codeur H.264/AVC et transmis sur un canal AWGN. Pour améliorer l'efficacité et réduire la complexité du décodage, des marqueurs de synchronisation ont été ajoutés dans le flux transmis. Les résultats de simulation ont démontré que le CRC avait un impact significatif sur les performances générales du traitement. Dans des conditions réalistes (avec codage canal), le gain en PSNR apporté par le CRC peut être supérieur à 3 dB. Par ailleurs, cette méthode implique des modifications à la fois dans le serveur et dans le terminal. L'encodeur du serveur doit générer les marqueurs de synchronisation et le terminal doit être capable d'intégrer le décodeur robuste à la couche APL (pile protocolaire perméable).

Le modèle en couche perméable permet d'intégrer les méthodes JSCD dans la couche APL du récepteur. Avec ce mécanisme, les informations souples fournies par le décodeur canal SISO de la couche PHY peuvent remonter jusqu'à la couche APL. Le principe repose sur la constatation suivante : les informations utilisées par une couche protocolaire sont placées dans l'en-tête des paquets respectifs. L'objectif de la couche perméable proposée consiste uniquement à corriger l'en-tête des paquets, sans prendre en considération les informations utiles transportées dans la *payload*. Le traitement de correction est basé sur une estimation au sens MAP qui exploite conjointement les redondances intra-couche et inter-couches ainsi que les propriétés du CRC. Ce mécanisme a été appliqué aux couches PHY et MAC du protocole WiFi. Les simulations ont montré que des gains de codage de 12 dB pouvaient être

atteints en modifiant légèrement le protocole de la couche MAC, c'est-à-dire en utilisant le CRC uniquement pour protéger l'en-tête (MAC-Lite). Par conséquent, une implémentation efficace de cette technique implique des modifications à la fois dans le terminal et dans le point d'accès (pour intégrer le protocole MAC-Lite).

Perspectives

Concernant les travaux sur le décodeur vidéo robuste (exploitant les techniques JSCD), il est impératif de réduire le sur-débit engendré par la transmission des marqueurs de synchronisation. Ces marqueurs indiquent actuellement la longueur binaire de chaque bloc de texture (résidus de prédiction) et la quantité additionnelle d'information à transmettre représente en moyenne 30 % du débit total. Pour réduire ce surcoût, une idée originale consisterait à ne transmettre que la position associée à chaque macrobloc. Des études préliminaires ont montré que cette technique permettrait de réduire le sur-débit d'un facteur 4, c'est-à-dire de le ramener à environ 8 % du débit total. Ce paramétrage pourrait donc représenter un bon compromis entre l'efficacité du décodage et le taux de redondance. Les performances et la complexité du décodage restent cependant à évaluer.

Les méthodes de décodage robuste, que nous avons présenté dans ce rapport, s'appuient sur un traitement séquentiel. Ce mécanisme permet de décoder itérativement une séquence complète en la divisant en portions élémentaires de quelques bits (voir partie 3.4 pour plus de précision). A chaque itération, les nouvelles combinaisons sont construites et l'algorithme ne conserve que les M chemins les plus probables. Ces derniers sont alors stockés dans une mémoire avant d'être réutilisés à la prochaine itération. A la dernière itération, seule la séquence la plus probable est sélectionnée. Nous nous sommes intéressés à cette dernière étape du traitement et nous avons pu constater que la séquence valide était rarement placée en première position de la mémoire quand le canal se dégradait. En revanche, elle faisait souvent partie des M combinaisons conservées. A titre d'exemple, nous avons réalisé une simulation dans laquelle l'algorithme comparait, à la dernière itération du décodage robuste, la séquence valide avec les cinq combinaisons les plus probables de la mémoire (à la place de tester uniquement la première combinaison). Nous avons ensuite reporté les résultats obtenus sur des graphiques représentant l'évolution de l'IBER (*Image Block Error Rate*) en fonction du SNR. Nous avons pu remarquer que cette méthode permettait d'atteindre des gains de codage de 7 dB par rapport aux techniques de décodage robuste standards. La deuxième piste à explorer découle donc de cette analyse. Son objectif viserait à sélectionner plus finement la combinaison finale lors de la dernière itération. Cette condition revient à redéfinir la métrique de sélection associée à la dernière étape du décodage robuste (en prenant en compte d'autres sources de redondance). Actuellement, l'idée que nous avons trouvé consiste à exploiter les informations visuelles contenues dans les blocs contigus déjà décodés. En somme, la technique consisterait à décoder les M combinaisons de la mémoire et à choisir le bloc le plus adapté au reste de l'image (comme dans un *puzzle* où les pièces sont carrées). Ce mécanisme correspondrait à un schéma JSCD combiné à un mécanisme d'*error concealment*.

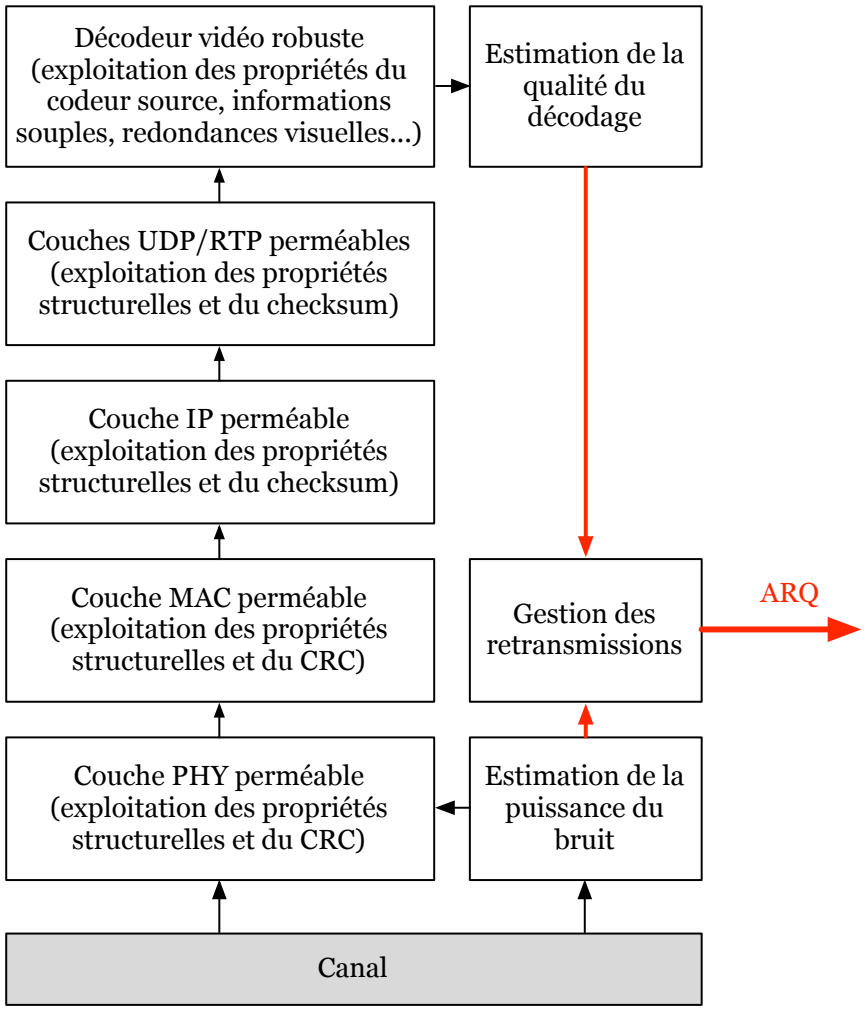


FIG. 4.11 – Schéma optimisé du récepteur pour la transmission robuste de vidéo

Concernant les travaux liés au modèle en couche perméable, il est nécessaire d'étendre le principe aux couches IP et UDP. Cette extension permettrait d'aboutir à une pile protocolaire totalement perméable. Des études réalisées sur le protocole ROHC ont démontré que les redondances temporelles entre les en-têtes des couches IP, UDP et RTP étaient significatives. A titre d'illustration, ce mécanisme de compression permet de compacter les 40 octets d'en-têtes en seulement quelques octets. Par ailleurs, dans notre contexte, la capacité de correction des en-têtes pourrait être renforcée en combinant les redondances structurelles aux informations apportées par le checksum. Notons simplement que le protocole UDP-Lite pourrait être utilisé dans la couche UDP pour améliorer les performances du décodage (comme à la couche MAC). Dans ce cas, le checkusm pourrait protéger les en-têtes UDP, RTP et éventuellement NAL.

Finalement, le dernier point à aborder concerne la gestion des retransmissions (quand elles sont possibles). Dans les travaux associés au mécanisme perméable (voir partie 4.5.6), nous

avons proposé un système de retransmission basé sur la qualité des informations souples (le CRC étant utilisé dans le décodage). Quand la puissance du signal est trop faible pour fournir une correction satisfaisante des en-têtes, le fragment MAC correspondant est retransmis par le point d'accès WiFi. La valeur du seuil de retransmission est déterminée de manière empirique : elle dépend de l'efficacité de correction de la couche la moins robuste. Au niveau du décodeur vidéo de la couche APL, le scénario est différent car chaque bit a une influence inégale sur la qualité de la vidéo décodée. Durant cette thèse, nous avons tenté de construire une métrique de retransmission basée sur une évaluation qualitative de la vidéo décodée. Cette métrique s'appuyait sur l'hypothèse de continuité des contours et de l'énergie dans les images pour détecter les artefacts trop importants. Cet estimateur n'était cependant pas efficace car il dépendait d'un seuil empirique qui ne s'adaptait pas au contenu de la vidéo. Actuellement, une autre méthode est en cours de développement et a déjà abouti à des résultats encourageants. Elle consiste à comparer les métriques associées aux M combinaisons générées à la dernière itération de la procédure de décodage robuste (en se basant sur un test d'hypothèse). Une retransmission est réalisée lorsque l'écart entre la métrique de la séquence candidate et la somme des métriques des autres combinaisons est trop faible. Le fragment retransmis pourrait ensuite être combiné à l'ancien fragment pour renforcer les performances du traitement robuste (HARQ ou *Hybrid ARQ*).

Le schéma général du récepteur, incorporant tous ces mécanismes, est illustré sur la figure 4.11. Ne manquent que... les résultats !

Références bibliographiques

[1] C. Reader. *History of MPEG Video Compression - Ver. 4.0.* Joint Video Team (JVT) doc, 2002. JVT-E066.

[2] ITU, CCITT. *Recommandation IT-81, Information Technology - Digital Compression and Coding of Continuous-Tone Still Images - Requirements and Guidelines (JPEG)*, 1992.

[3] R. D. Kell. Improvements relating to electric picture transmission systems. Technical report, British patent No. 341.811, 1929.

[4] ITU-T. Codec for videoconferencing using primary digital group management. Technical report, ITU-T Rec. H.120, version 1, 1984.

[5] A. Habibi. Hybrid coding of pictorial data. *IEEE Trans. on Communications*, 22(5) :614–624, 1974.

[6] ITU-T. Video codec for audiovisual services at px64 kbits/s. Technical report, ITU-T Rec. H.261, version 1, nov. 1990.

[7] ISO/IEC JTC 1. Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbis/s - part 2 : Video. Technical report, ISO/IEC 11172-2 (MPEG-1), mar. 1993.

[8] ISO/IEC JTC 1/SC 29. Generic coding of moving pictures and associated audio information : Systems. Technical report, ISO/IEC 13818-1 (MPEG-2 Part 1), 1996.

[9] ITU-T. Video coding for low bit rate communication. Technical report, ITU-T Rec. H.263, version 1, nov. 1995.

[10] ISO/IEC JTC 1. Coding of audio-visual objects - part 2 : Video. Technical report, ISO/IEC 14496-2 (MPEG-4 visual version 1), apr. 1999.

[11] ITU-T and ISO/IEC JTC 1. Advanced video coding for generic audiovisual services. Technical report, ITU-T Rec. H.264, and ISO/IEC 14496-10 AVC, nov. 2003.

[12] K. Sayood. *Introduction to Data Compression, Second Edition*. Morgan Kaufmann, San Francisco, 2000.

[13] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[14] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Trans. on Circuits and Systems for Video Technology*, 13(7) :560–576, 2003.

[15] ITU. Methodology for the subjective assessment of the quality of television pictures. Technical report, ITU-R BT. 500-11, 2004.

[16] I. Richardson. *H.264 and MPEG-4 Video Compression : Video Coding for Next-Generation Multimedia*. John Wiley and Sons, 2003.

[17] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky. Low-complexity transform and quantization in H.264/AVC. *IEEE Trans. on Circuits and Systems for Video Technology*, 13(7) :598– 603, 2003.

[18] G. Bjontegaard and K. Lillevold. Context-adaptive VLC coding of coefficients. Technical report, JVT, 2002.

[19] D. Marpe, H. Schwarz, and T. Weigand. Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Trans. on Circuits and Systems for Video Technology*, 13(7) :620–636, 2003.

[20] D. Marpe, G. Blattermann, and T. Wiegand. Adaptive codes for H.26L. Technical report, JVT, 2001.

[21] P. List, A. Joch, J. Lainema, G. Bjøntegaard, and M. Karczewicz. Adaptive deblocking filter. *IEEE Trans. on Circuits and Systems for Video Technology*, 13(7) :614–619, 2003.

[22] D. Hardy, G. Malléus, and J.-N. Méreur. *Réseaux : Internet, téléphonie, multimédia*. De Boeck Université, 2002.

[23] X. Lagrange, P. Godlewski, and S. Tabbane. *Réseaux GSM-DCS des principes à la norme*. Hermès science, 1999.

[24] R. J. Bates. *GPRS : General Packet Radio Service*. McGraw-Hill Professional, 2002.

[25] P. Lescuyer and P. Rosé. *UMTS*. Dunod, 2001.

[26] J. G. Andrews, A. Ghosh, and R. Muhamed. *Fundamentals of WiMAX : Understanding Broadband Wireless Networking*. Prentice Hall, 2007.

[27] U. Reimers. *DVB : The Family of International Standards for Digital Video Broadcasting*. Springer, 2005.

[28] R. Morrow. *Bluetooth : Operation and Use*. McGraw-Hill Professional, 2002.

[29] M. Gast. *802.11 Wireless Networks : The Definitive Guide*. O'Reilly, 2005.

[30] IEEE 802.16 standard. Local and metropolitan area networks part 16 : Air interface for fixed broadband wireless access systems amendment for physical and medium access control layers for combined fixed and mobile in license bands. Technical report, IEEE, 2004.

[31] IEEE 802.16e amendment. Physical and medium access control layers for combined fixed and mobile operation in license bands. Technical report, IEEE, 2005.

[32] J. Postel. Internet protocol. Technical Report RFC 791, Information Sciences Institute, 1981.

[33] J. Postel. Transport control protocol. Technical Report RFC 793, Information Sciences Institute, 1981.

[34] J. Postel. User datagram protocol. Technical Report RFC 768, Information Sciences Institute, 1980.

[35] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. A transport protocol for real-time applications. Technical Report RFC 1889, Network Working Group, 1996.

[36] ANSI/IEEE. 802.11, part 11 : Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Technical report, 1999.

[37] IEEE 802.2 standard. Local and metropolitan area networks part 2 : Logical link control. Technical report, IEEE, 1998.

[38] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, 2003.

[39] T. Richardson and U. Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008.

[40] P. Salama, N. Shroff, E. J. Coyle, and E. J. Delp. Error concealment techniques for encoded video streams. In *Proc. of ICIP*, pages 9–12, 1995.

[41] W. Zeng and B. Liu. Geometric-structure-based error concealment with novel applications in block-based low-bit-rate coding. *IEEE Trans. on Circuits and Systems for Video Technology*, 9(4) :648–665, 1999.

[42] H. Sun and W. Kwok. Concealment of damaged block transform coded images using porjections onto convex sets. *IEEE Trans. on Image Processing*, 4 :470–477, 1995.

[43] M.-J. Chen, L.-G. Chen, and R.-M. Weng. Error concealment of lost motion vectors with overlapped motion compensation. *IEEE Trans. on Circuits and Systems for Video Technology*, 7(3) :560–563, 1997.

[44] E. Asbun and E. J. Delp. Real-time error concealment in compressed digital video streams. In *Proc. of PCS*, 1999.

[45] Y. O. Park, C.-S. Kim, and S.-U. Lee. Multi-hypothesis error concealment algorithm for H.26L video. In *Proc. of ICIP*, pages 465–468, 2003.

[46] B. Jung, B. Jeon, M.-D. Kim, B. Suh, and S.-I. Choi. Selective temporal error concealment algorithm for H.264/AVC. In *Proc. of ICIP*, 2004.

[47] Y.-K. Wang, M. M. Hannuksela, V. Varsa, A. Hounrunranta, and M. Gabbouj. The error concealment feature in the H.26L test model. In *Proc. of ICIP*, volume 2, pages 729–732, 2002.

[48] H. Sun and J. Zedepski. Adaptive error concealment algorithm for MPEG compressed video. In *Proc. of VCIP*, pages 814–824, 1992.

[49] W.-Y. Kung, C.-S. Kim, and C.-C. J. Kuo. Spatial and temporal error concealment techniques for video transmission over noisy channels. *IEEE Trans. on Circuits and Systems for Video Technology*, 16 :789–802, 2006.

[50] V. Buttigieg and P.G. Farrell. A MAP decoding algorithm for variable-length error-correcting codes. In *Codes and Cyphers : Cryptography and Coding IV*, pages 103–119, Essex, England, 1995. The Inst. of Mathematics and its Appl.

[51] J. Hagenauer. Source-controlled channel decoding. *IEEE Trans. on Communications*, 43(9) :2449–2457, 1995.

[52] S. Kaiser and M. Bystrom. Soft decoding of variable-length codes. In *Proc. of ICC*, volume 3, pages 1203–1207, New Orleans, 2000.

[53] L. Perros-Meilhac and C. Lamy. Huffman tree based metric derivation for a low-complexity soft VLC decoding. In *Proc. of ICC*, 2002.

[54] R. Thobanen and J. Kliewer. Robust decoding of variable-length encoded markov sources using a three-dimensional trellis. *IEEE Communications Letters*, 7(7) :320–322, 2003.

[55] T. Tillo, M. Grangetto, and G. Olmo. A flexible error resilient scheme for JPEG 2000. In *Proc. of MSP*, pages 295–298, 29 Sept.-1 Oct. 2004.

[56] H. Nguyen, P. Duhamel, J. Brouet, and D. Rouffet. Robust VLC sequence decoding exploiting additional video stream properties with reduced complexity. In *Proc. of ICME*, pages 375–378, June 2004. Taipei, Taiwan.

[57] C. Bergeron and C. Lamy-Bergot. Soft-input decoding of variable-length codes applied to the H.264 standard. In *Proc. of MSP*, pages 87–90, 29 Sept.-1 Oct. 2004.

[58] G. Sabeva, S. Ben Jamaa, M. Kieffer, and P. Duhamel. Robust decoding of H.264 encoded video transmitted over wireless channels. In *Proc. of MSP*, pages 9–12, Victoria, Canada, 2006.

[59] C.M. Lee, M. Kieffer, and P. Duhamel. Soft decoding of VLC encoded data for robust transmission of packetized video. In *Proc. of ICASSP*, pages 737–740, 2005.

[60] R. Bauer and J. Hagenauer. On variable length codes for iterative source/channel decoding. In *Proc. of DCC*, pages 272–282, Snowbird, UT, 1998.

[61] R. Thobaben and J. Kliewer. On iterative source-channel decoding for variable-length encoded markov sources using a bit-level trellis. In *Proc. of SPAWC*, Rome, 2003.

[62] H. Nguyen and P. Duhamel. Iterative joint source-channel decoding of variable length encoded video sequences exploiting source semantics. In *Proc. of ICIP*, 2004.

[63] C. Lamy and S. Merigeault. Procédé de correction d'une trame erronée par un récepteur. French patent no. 0206501, 2002.

[64] J. W. Nieto and W. N. Furman. Cyclic redundancy check (CRC) based error correction method and device. US Patent 0192667 A1, Aug. 16 2007.

[65] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. on Information Theory*, 20 :284–287, 1974.

[66] J. K. Wolf. Efficient maximum-likelihood decoding of linear block codes using a trellis. *IEEE Trans. on Information Theory*, 24 :76–80, 1978.

[67] J. B. Anderson and S. Mohan. *Source and Channel Coding : An Algorithmic Approach*. Kluwer, 1991.

[68] R. E. Blahut. *Theory and Practice of Error Control Codes*. Addison-Wesley, Reading, MA, 1984.

[69] F. Fontaine and S. B. Wicker. Method ans apparatus for turbo decoding block codes. US Patent 7243288 B1, Jul. 10 2007.

[70] L. A. Larzon, M. Degermark, L. E. Jonsson, and G. Fairhurst. The lightweight user datagram protocol (UDP-Lite). Technical Report RFC 3828, The Internet Society, 2004.

[71] H. Jenka, T. Stockhammer, and W. Xu. Permeable-layer receiver for reliable multicast transmission in wireless systems. In *Proc. of WCNC*, volume 3, pages 1805–1811Vol.3, 13-17 March 2005.

[72] H. Jenkac, T. Stockhammer, and W. Xu. Cross-layer assisted reliability design for wireless multimedia broadcast. *EURASIP Signal Processing Journal*, 2006.

[73] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L.-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng. Robust header compression (ROHC) : Framework and four profiles, 2001.

Annexes

Annexe A
Encodage CAVLC des résidus de prédiction

Cette annexe fournit une description précise de la méthode CAVLC lors de l'encodage des résidus de prédiction. Ces résidus représentent les coefficients quantifiés en fréquence sortant de l'étage de transformation. Ils sont généralement rassemblés dans des blocs de 4×4 composantes, mais peuvent aussi correspondre à des blocs de 2×2 coefficients dans le cas des signaux de chrominance DC. Cette technique a été conçue pour exploiter les caractéristiques énumérées ci-dessous :

- Après les étapes de prédiction, de transformation et de quantification, les blocs sont particulièrement clairsemés (ils contiennent de nombreux coefficients nuls situés essentiellement dans les hautes fréquences). Le CAVLC exploite cette singularité en regroupant les séquences de zéros à la fin d'un tableau unidimensionnel. Pour réaliser cette opération, il répartit les coefficients matriciels dans un tableau. Il balaie la matrice en respectant un modèle particulier (*zig-zag scan*), c'est-à-dire en partant des basses fréquences pour atteindre les hautes fréquences.
- Les études ont montré que les derniers coefficients non-nuls du tableau étaient souvent des séquences de $+/-1$. Ces éléments sont appelés T1s. Le paramètre *TrailingOnes* définit le nombre de T1s. La technique de codage tire partie de cette propriété en encodant des coefficients de manière particulière.
- Généralement, il existe une corrélation importante entre les blocs contigus contenant les résidus de prédiction. Le codeur utilise cette caractéristique pour sélectionner la table VLC la plus adaptée au codage de l'élément *TotalCoeffs*, qui représente le nombre de coefficients non-nuls du bloc courant.
- L'expérience pratique a montré que l'amplitude des coefficients était plus élevée au début du tableau (vers les basses fréquences) et plus faible dans les hautes fréquences. Le CAVLC utilise cette propriété lors de l'encodage de la valeur des coefficients non-nuls (appelée *Levels*)

L'encodage des blocs s'exécute en plusieurs étapes qui sont listées ci-dessous.

1 Le codage de *TotalCoeffs* et de *TrailingOnes*

Le premier paramètre, *CoeffToken*, rassemble dans un simple mot de code les éléments *TotalCoeffs* et *TrailingOnes*.

Lors de l'encodage d'un bloc 4×4, le paramètre *TotalCoeffs*, qui représente le nombre de coefficients non-nuls, peut prendre une valeur comprise entre 0 et 16. Par ailleurs, le paramètre *TrailingOnes*, qui représente le nombre de $+/-1$ situés à la fin du tableau, possède une valeur variant entre 0 et 3. Dans le cas où *TrailingOnes* est supérieur à 3, seuls les trois derniers coefficients sont traités d'une manière particulière. Les autres sont codés comme des coefficients standards.

Les mots de code, utilisés pour encoder *CoeffToken*, sont extraits d'une des quatre tables VLC exposées sur la figure 1. La sélection dépend de certaines caractéristiques propres aux blocs contigus. Plus précisément, le choix s'effectue en fonction du nombre de coefficients

non-nuls contenus dans les blocs situés au-dessus (N_U) et à gauche (N_L) du bloc courant. Le critère, nommé N, est calculé de la façon suivante :

- Si les blocs U et L sont tous les deux disponibles, $N = (N_U + N_L)/2$.
- Si seul le bloc U est disponible, $N = N_U$.
- Si seul le bloc L est disponible, $N = N_L$.
- Dans tous les autres cas, $N = 0$.

Le paramètre N permet de sélectionner la table de codage appropriée et illustre le modèle adaptatif de la méthode. La première table est biaisée vers les indices faibles, la deuxième vers les valeurs intermédiaires et la troisième à proximité des nombres importants. Par conséquent, la corrélation entre les blocs est astucieusement exploitée et permet d'appréhender la valeur la plus probable afin d'assigner le mot de code VLC le plus court au paramètre à encoder. La dernière table est composée de mots de code de longueur fixe. Pour résumer, trois paramètres permettent de définir un mot de code : la table adaptative est désignée en fonction de N et la ligne est déterminée à partir de *TotalCoeffs* et de *TrailingOnes*.

En ce qui concerne l'encodage des blocs 2×2, la méthode se limite à une seule table VLC (dernière colonne de la figure 1). Les valeurs de *TotalCoeffs* sont alors comprises entre 0 et 4.

2 Le codage des signes de *TrailingOnes*

Un simple bit permet de coder le signe de chaque T1. Ainsi, un niveau logique égale à 1 correspond à un signe négatif (-1), et dans le cas contraire, à une valeur positive ($+1$). Ces éléments sont encodés en respectant un ordre de balayage inverse, c'est-à-dire en partant des basses fréquences pour remonter vers les hautes fréquences.

3 Le codage des *Levels*

Les *Levels* (amplitudes et signes des coefficients non-nuls restants) sont également encodés en respectant un ordre de balayage inverse. L'amplitude et le signe de chaque coefficient sont incorporés dans le paramètre *Code* en suivant la procédure définie par (1) et (2). Dans ces équations, $\ll$ réalise un décalage binaire de 1 bit vers la gauche et $\cup$ représente l'opérateur OR (*ou logique*). Le signe est codé sur le bit de poids faible et l'amplitude sur les bits de poids forts.

$$Code = (amplitude - 1) \ll 1 \tag{1}$$

$$Code = Code \cup signe \tag{2}$$

Le paramètre *Code* est ensuite décomposé en deux éléments de syntaxe : *LevelPrefix* et *LevelSuffix*. Les valeurs numériques de ces éléments sont déterminées à partir de (3) et (4). Dans ces équations, $\gg$ réalise un décalage de 1 bit vers la droite et $Code(Shift)$ représente

TrailingOnes	*TotalCoeffs*	$0 \leqslant N < 2$	$2 \leqslant N < 4$	$4 \leqslant N < 8$	$N \geqslant 8$	$N = -1$
0	0	1	11	1111	000011	01
0	1	000101	001011	001111	000000	000111
1	1	01	10	1110	000001	1
0	2	00000111	000111	001011	000100	000100
1	2	000100	00111	01111	000101	000110
2	2	001	011	1101	000110	001
0	3	0000000111	000011	001000	001000	000011
1	3	00000110	001010	01100	001001	0000011
2	3	0000101	001001	01110	001010	0000010
3	3	00011	0101	1100	001011	000101
0	4	0000000111	00000111	0001111	001100	000010
1	4	000000110	000110	01010	001101	00000011
2	4	00000101	000101	01011	001110	00000010
3	4	000011	0100	1011	001111	0000000
0	5	00000000111	00000100	0001011	010000	-
1	5	0000000110	0000110	01000	010001	-
2	5	000000101	0000101	01001	010010	-
3	5	0000100	00110	1010	010011	-
0	6	0000000001111	000000111	0001001	010100	-
1	6	00000000110	00000110	001110	010101	-
2	6	0000000101	00000101	001101	010110	-
3	6	00000100	001000	1001	010111	-
0	7	0000000001011	00000001111	0001000	011000	-
1	7	0000000001110	000000110	001010	011001	-
2	7	00000000101	000000101	001001	011010	-
3	7	000000100	000100	1000	011011	-
0	8	0000000001000	00000001011	00001111	011100	-
1	8	0000000001010	00000001110	0001110	011101	-
2	8	0000000001101	00000001101	0001101	011110	-
3	8	0000000100	0000100	01101	011111	-
0	9	00000000001111	000000001111	00001011	100000	-
1	9	00000000001110	00000001010	00001110	100001	-
2	9	0000000001001	00000001001	0001010	100010	-
3	9	00000000100	000000100	001100	100011	-
0	10	00000000001011	000000001011	000001111	100100	-
1	10	00000000001010	000000001110	00001010	100101	-
2	10	00000000001101	000000001101	00001101	100110	-
3	10	0000000001100	00000001100	0001100	100111	-
0	11	000000000001111	000000001000	000001011	101000	-
1	11	000000000001110	000000001010	000001110	101001	-
2	11	00000000001001	000000001001	00001001	101010	-
3	11	00000000001100	00000001000	00001100	101011	-
0	12	000000000001011	0000000001111	000001000	101100	-
1	12	000000000001010	0000000001110	000001010	101101	-
2	12	000000000001101	0000000001101	000001101	101110	-
3	12	00000000001000	000000001100	00001000	101111	-
0	13	0000000000001111	0000000001011	0000001101	110000	-
1	13	000000000000001	0000000001010	000000111	110001	-
2	13	000000000001001	0000000001001	000001001	110010	-
3	13	000000000001100	0000000001100	000001100	110011	-
0	14	0000000000001011	0000000000111	0000001001	110100	-
1	14	0000000000001110	00000000001011	0000001100	110101	-
2	14	0000000000001101	0000000000110	0000001011	110110	-
3	14	000000000001000	0000000001000	0000001010	110111	-
0	15	0000000000000111	00000000001001	0000000101	111000	-
1	15	0000000000001010	00000000001000	0000001000	111001	-
2	15	0000000000001001	00000000001010	0000000111	111010	-
3	15	0000000000001100	0000000000001	0000000110	111011	-
0	16	0000000000000100	00000000000111	0000000001	111100	-
1	16	0000000000000110	00000000000110	0000000100	111101	-
2	16	0000000000000101	00000000000101	0000000011	111110	-
3	16	0000000000001000	00000000000100	0000000010	111111	-

FIG. 1 – Tables VLC utilisées pour l'encodage de *TotalCoeffs*

la valeur correspondant aux *Shift* bits de poids faibles de *Code*.

$$LevelPrefix = Code \gg Shift \tag{3}$$

$$LevelSuffix = Code(Shift) \tag{4}$$

Finalement, l'élément de syntaxe *LevelPrefix* est encodé à l'aide de la table VLC exposée sur la figure 2 et les *Shift* bits de *LevelSuffix* sont ajoutés à la suite. Le principe d'adaptabilité dans la phase d'encodage des *Levels* repose essentiellement sur la variable *Shift*. Cette valeur peut évoluer au cours de la phase d'encodage des *Levels*. Pour le premier coefficient, elle est initialisée à 0. Pour les cycles suivants, elle est incrémentée de 1 si l'amplitude du coefficient courant (déjà encodé) est supérieure au seuil associé à la valeur courante de *Shift*, défini dans la table de la figure 3. Cette technique permet à l'encodeur de s'ajuster au mieux à la dynamique des coefficients restant à encoder. Sachant que l'évolution des amplitudes a tendance à croître lorsque la fréquence diminue, cette technique s'avère très efficace dans cette situation.

LevelPrefix	Mot de code
0	1
1	01
2	001
3	0001
4	00001
5	000001
6	0000001
7	00000001
8	000000001
9	0000000001
10	00000000001
11	000000000001
12	0000000000001
13	00000000000001
14	000000000000001
15	0000000000000001

FIG. 2 – Table VLC utilisée pour l'encodage de *LevelPrefix*

L'opération de décodage est composée de plusieurs étapes. Dans un premier temps, le décodeur commence par récupérer *LevelPrefix*. Il extrait ensuite l'élément *LevelSuffix* du flux en se basant sur la valeur courante de *Shift*. En inversant les équations (1), (2), (3) et (4), le décodeur arrive à retrouver l'amplitude et le signe associés à chaque coefficient. Par ailleurs, il met à jour la valeur de *Shift* à chaque itération.

Shift	Valeur du seuil
0	0
1	3
2	6
3	12
4	24
5	48
6	∞

FIG. 3 – Valeurs des seuils associés à la variable *Shift*

4 Le codage de *TotalZeros*

Le paramètre *TotalZeros* définit le nombre de zéros précédant le dernier coefficient non-nul dans le tableau ordonné. Pour les blocs 4×4, cet élément est encodé en utilisant les tables VLC représentées sur la figure 4. Les tables définies sur la figure 5 sont appliquées pour les blocs 2×2. Dans les deux cas, la table est sélectionnée en fonction de la valeur de *TotalCoeffs*. Le nombre de possibilités décroît avec l'augmentation du nombre de coefficients non-nuls dans le bloc. Les mots de code VLC contenus dans ces tables ont été obtenus de façon empirique et le lecteur peut remarquer les disparités existantes entre deux tables voisines (singularité du biais).

5 Le codage des *RunBefores*

Le nombre de zéros précédant chaque coefficient non-nul (*Runbefore*) est encodé dans l'ordre de balayage inverse. Un élément *RunBefore* est associé à chaque coefficient non-nul. Il existe cependant deux exceptions :

1. Si les *TotalZeros* coefficients nuls ont été parcourus, il n'est pas nécessaire d'envoyer plus d'information.
2. Il n'est pas indispensable de coder le paramètre *RunBefore* correspondant au dernier coefficient (composante DC).

Les tables utilisées pour encoder *RunBefores* sont illustrées sur la figure 6. La table courante est sélectionnée en fonction du nombre de zéros qui n'ont pas encore été parcourus, défini par la variable *ZerosLeft*. *ZerosLeft* est mis à jour à chaque itération et représente le paramètre d'adaptation de l'étape d'encodage des *RunBefores*.

Exemple 1 *Ce paragraphe fournit un exemple simple qui permet de mieux appréhender les différentes phases d'encodage d'un bloc. Nous faisons l'hypothèse que la première table de la figure 1 est utilisée pour encoder CoeffToken, c'est-à-dire que* $0 \leqslant N < 2$.

Considérons le bloc 4×4 *suivant :*

0	*3*	*-1*	*0*
0	*-1*	*1*	*0*
1	*0*	*0*	*0*
0	*0*	*0*	*0*

Les coefficients du bloc sont ensuite répartis dans un tableau unidimensionnel en suivant une procédure de balayage particulier : des basses fréquences vers les hautes fréquences. A la suite du traitement, on aboutit à la représentation suivante :

0	*3*	*0*	*1*	*-1*	*-1*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*

Ce tableau est utilisé lors de la procédure d'encodage. Le codeur commence par extraire les paramètres fondamentaux qui ont été étudiés dans ce chapitre :

- *Le nombre de coefficients non-nuls (*$TotalCoeffs = 5$*).*
- *La nombre de zéros précédant le dernier coefficient non-nul (*$TotalZeros = 3$*).*
- *Le nombre de coefficients égaux à* $+/-1$ *situés à la fin du tableau (*$TrailingOnes = 3$ *car le nombre maximal est limitée à* 3*).*

Le traitement se poursuit et permet d'aboutir aux résultats suivant :

Elément	*Valeur*	*Code*	*Référence*
CoeffToken	$TotalCoeffs = 5$, $TrailingOnes = 3$	*0000100*	*Fig. 1 - Tab.* 1
Signe T1 (4)	$+$	*0*	
Signe T1 (3)	$-$	*1*	
Signe T1 (2)	$-$	*1*	
Level (1)	*+1 (*$Shift = 0$*)*	*1*	*Fig. 2 et 3*
Level (0)	*+3 (*$Shift = 1$*)*	*0010*	*Fig. 2*
TotalZeros	*3*	*111*	*Fig. 4 - Tab.* 5
RunBefore (4)	$ZerosLeft = 3$, $RunBefore = 1$	*10*	*Fig. 6 - Tab.* 3
RunBefore (3)	$ZerosLeft = 2$, $RunBefore = 0$	*1*	*Fig. 6 - Tab.* 2
RunBefore (2)	$ZerosLeft = 2$, $RunBefore = 0$	*1*	*Fig. 6 - Tab.* 2
RunBefore (1)	$ZerosLeft = 2$, $RunBefore = 1$	*01*	*Fig. 6 - Tab.* 2
RunBefore (0)	$ZerosLeft = 1$, $RunBefore = 1$	*Pas utile*	*Dernier coefficient*

Par conséquent, la séquence transmise pour ce bloc est :

0	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*1*	*1*	*0*	*0*	*1*	*0*	*1*	*1*	*1*	*1*	*0*	*1*	*1*	*0*	*1*

TotalZeros	*TotalCoeffs*						
	1	2	3	4	5	6	7
0	1	111	0101	00011	0101	000001	000001
1	011	110	111	111	0100	00001	00001
2	010	101	110	0101	0011	111	101
3	0011	100	101	0100	111	110	100
4	0010	011	0100	110	110	101	011
5	00011	0101	0011	101	101	100	11
6	00010	0100	100	100	100	011	010
7	000011	0011	011	0011	011	010	0001
8	000010	0010	0010	011	0010	0001	001
9	0000011	00011	00011	0010	00001	001	000000
10	0000010	00010	00010	00010	0001	000000	-
11	00000011	000011	000001	00001	00000	-	-
12	00000010	000010	00001	00000	-	-	-
13	000000011	000001	000000	-	-	-	-
14	000000010	000000	-	-	-	-	-
15	000000001	-	-	-	-	-	-

TotalZeros	*TotalCoeffs*							
	8	9	10	11	12	13	14	15
0	000001	000001	00001	0000	0000	000	00	0
1	0001	000000	00000	0001	0001	001	01	1
2	00001	0001	001	001	01	1	1	-
3	011	11	11	010	1	01	-	-
4	11	10	10	1	001	-	-	-
5	10	001	01	011	-	-	-	-
6	010	01	0001	-	-	-	-	-
7	001	00001	-	-	-	-	-	-
8	000000	-	-	-	-	-	-	-

FIG. 4 – Tables VLC utilisées pour l'encodage de *TotalZeros* dans les blocs 4×4

TotalZeros	*TotalCoeffs*		
	1	2	3
0	1	1	1
1	01	01	0
2	001	00	-
3	000	-	-

FIG. 5 – Tables VLC utilisée pour l'encodage de *TotalZeros* dans les blocs 2×2

RunBefore	*ZerosLeft*						
	1	2	3	4	5	6	> 6
0	1	1	11	11	11	11	111
1	0	01	10	10	10	000	110
2	-	00	01	01	011	001	101
3	-	-	00	001	010	011	100
4	-	-	-	000	001	010	011
5	-	-	-	-	000	101	010
6	-	-	-	-	-	100	001
7	-	-	-	-	-	-	0001
8	-	-	-	-	-	-	00001
9	-	-	-	-	-	-	000001
10	-	-	-	-	-	-	0000001
11	-	-	-	-	-	-	00000001
12	-	-	-	-	-	-	000000001
13	-	-	-	-	-	-	0000000001
14	-	-	-	-	-	-	00000000001

FIG. 6 – Tables VLC utilisées pour le codage des *RunBefores*

Annexe B
Premier article IEEE

Joint Exploitation of Residual Source Information and MAC Layer CRC Redundancy for Robust Video Decoding

Cédric Marin, Khaled Bouchireb, Michel Kieffer, *Senior Member, IEEE*, and Pierre Duhamel, *Fellow, IEEE*

Abstract

This paper presents a MAP estimation method for the robust decoding of compressed video stream. A sequential decoding algorithm jointly exploiting the residual source information (related to VLC codes and source characteristics) along with the MAC layer CRC redundancy in the transmission scheme is proposed. The branch selection metric in the decoding trellis incorporates the usually considered APP weighted by a CRC dependent factor. We also introduce a suboptimal but hardware realizable version of the proposed algorithm. This technique has been applied to the robust decoding of CAVLC encoded sequences in the H.264/AVC standard. Significant link budget improvement results have been demonstrated for transmission schemes using BPSK modulated signals sent over AWGN channels.

Index Terms

Soft decoding of linear block codes, sequential decoding, source decoding based on syntax and semantic of bitstream, WiFi, H.264

I. INTRODUCTION

The highly difficult wireless channel presents a major challenge for high bitrate transmission. Factors such as high signal attenuation, multiple access interference, inter-symbol interference, and Doppler shift can heavily degrade signal quality. Consequently, the typical BER encountered

C. Marin is with the Research and Innovation Center of Bell Labs Alcatel-Lucent, France.

K. Bouchireb, M. Kieffer, and P. Duhamel are with the L2S of CNRS, France.

in mobile transmission can be several orders of magnitude higher than in wire line (*e.g.*, DSL) transmission.

High efficiency video transmission is usually dependent on compression mechanism applied to the image stream [25]. Nevertheless, the compressed video flow is very sensitive to transmission errors. A single error can lead to a decoder de-synchronization resulting in a total loss of remaining picture information or to inter-image error propagation due to inter-picture coding. Consequently, the video stream incoming in the video decoder has to be nearly error-free.

In wireless transmission, the received signal may be heavily corrupted and is not directly useable by the video decoder. A first solution to alleviate this problem consists in grouping data into packets protected by an error-detection code (CRC or checksum) [5], [13]. Packets which integrity is not ensured at receiver side may then be retransmitted. Nevertheless, retransmissions may become difficult in scenarii with strong delay constraints (*e.g.*, for visiophony), or even impossible when broadcasting data (*e.g.*, in satellite television).

In such situations, the standard solutions perform channel decoding processes by exploiting very strong error-correction codes (*e.g.*, turbo codes, LDPC) at *Physical* (PHY) layer combined with packet-erasure codes (*e.g.*, Reed-Solomon) at intermediate protocol layers [16], [23]. Nevertheless, redundancy is rarely optimally dimensioned. It may be oversized when the channel is clear, reducing the bandwidth allocated for the data. In contrary, some corrupted packets could not be recovered in bad channel conditions, leading to their lost. Error-concealment techniques [8], [12] may then be used by the source decoders at *Application* (APL) layer. They exploit the redundancy (temporal and/or spatial) in the decoded multimedia stream for estimating the missing information.

In the last years, joint source-channel decoding techniques have been proposed to correct damaged packets. These methods involve robust source decoders, which exploit the inherent redundancy in the received packets, for correcting errors. Several sources of redundancy have been identified. Constraints in the syntax of variable-length codes [6], [7], [11], [21], [27] have been used first. Then, the properties due to the semantic of the source coders have been combined along with the syntax redundancy to improve the performance of robust decoders [4], [19], [24], [28]. Redundancy associated to the packetization of coded data have been introduced in [15]. Recently, information introduced by the channel codes have been jointly employed together with the residual redundancy through iterative decoding processes [3], [18], [26]. These joint schemes

improve the decoding performance when compared to classical schemes.

This paper focuses on robust decoding of video data in a downlink situation. We propose a sequential decoding algorithm jointly exploiting the syntax and semantic properties of coded video stream along with the redundancy at MAC layer provided by the CRC. The CRC is not used to detect errors but is considered as an error correcting code. This CRC based decoding approach has been presented in [14], [20] for correcting erroneous packets. We propose here to make usage of both the CRC and the source redundancy to improve the video decoding performance. This paper is based on a variety of techniques (soft decoding of block codes [2], sequential decoding [1], source decoding depending on syntax and semantic of bitstream [4]) which are combined to attain our objective.

All the robust techniques introduced above require soft information to be delivered from the PHY layer to the APL layer. This may be achieved by using an header recovery techniques exploiting the intra- and inter-layer redundancies along with the CRCs or checksums as in [17]. With a such techniques, if the header is correctly decoded, the payload may be forwarded to the upper layers, resulting in a *permeable* protocol stack.

This paper is organized as follows. After a brief introduction of the proposed permeable protocol stack in Section II, Section III describes the derivation of the decoding metric and proposes a general sequential decoding method. Reduction of complexity is presented in Section IV. Finally, the simulation results are described in Section V before drawing some conclusions.

II. Model of Permeable Protocol Stack

Multimedia packetized transmission usually relies on a multi-layer architecture [13] based on the RTP/UDP/IP stack.

Figure 1 illustrates an example of the segmentation and encapsulation mechanisms implemented at each protocol layer in the case of video transmitted with a WiFi radio interface [9] (802.11 standard). The data processed by the PHY layer are forwarded to the MAC layer which checks its integrity with the help of CRC. For corrupted packets, a retransmission is requested. Correctly received data are assembled to form the binary stream that is then fed to the video decoder (at APL layer) after removal of IP, UDP, and RTP protocol headers.

A new protocol stack design where the PHY, MAC, and APL layers of the receiver work very closely together is presented here. Three changes are required to implement the proposed

solution:

- The PHY layer has to include a SISO (Soft-Input Soft-Output) channel decoder for processing the incoming protected data. The soft information are transmitted to the next layer.
- In the MAC layer, the CRC checking operation is deactivated and no retransmission are allowed. Complete MAC packets (composed of header, payload, and CRC) are transfered to the upper layer for being integrated in the payload of IP packets.
- The MAC header and CRC, usually not transmitted by the IP, UDP, and RTP layers, are assumed to be available at the APL layer.

These changes are facilitated by using the robust header recovery and permeable layer mechanisms presented in [17]. Hence, we can assumed that the headers are available without errors at all layers.

With these modifications, the APL layer receives a succession of MAC packets, containing soft information (provided by the PHY layer). These modifications only affect the receiver (the transmitter operations and the signal sent are unchanged). The format of data received by the APL layer is represented in Figure 2.

Additionally, the computational complexity can be minimized by deactivating the robust decoding processing when:

1) normal CRC check is successful,
2) the quality of soft information provided by the lower layer is too poor, *i.e.*, when the signal power is inferior to a pre-defined threshold. In such a case, the packet is retransmitted or discarded.

The next section presents the analytical derivation of the decoding metric which may be used to robustly reconstruct the transmitted video sequence. We then propose a sequential decoding algorithm based on this metric.

III. Group-Based Sequential Decoding

A. Notations

The symbols produced by all the components of a video coder before the entropy coding stage are assumed to be generated by a source $\mathcal{S}$, which has to satisfy some semantic rules. Consider a vector $\mathbf{m} = [m_1 \dots m_K]$ of K symbols generated by this source. The entropy coder associates

a variable-length codeword $\mathbf{x}_{m_i}$ to each component m_i of $\mathbf{m}$, $i = 1 \dots K$, which is then mapped onto a binary sequence $\mathbf{x} = [\mathbf{x}_{m_1} \dots \mathbf{x}_{m_K}]$, with

$$\sum_{i=1}^{K} \ell(\mathbf{x}_{m_i}) = \ell(\mathbf{x}). \tag{1}$$

In (1) and in what follows, $\ell(\mathbf{v})$ denotes the length in bits of the vector $\mathbf{v}$. Thus, $\mathbf{x}$ has to be compliant with the syntax of the variable-length code (VLC) and with the semantic rules of the source $\mathcal{S}$.

At the MAC layer, a header $\mathbf{h}$ is put in front of $\mathbf{x}$ to get a concatenated vector $\mathbf{d} = [\mathbf{h}, \mathbf{x}]$. A CRC $\mathbf{c}$ is then computed from the data $\mathbf{d}$ and added at the end of the packet. This set of information is collected in a vector $\mathbf{t} = [\mathbf{h}, \mathbf{x}, \mathbf{c}] = [\mathbf{d}, \mathbf{c}]$, where $\mathbf{c} = \mathcal{F}(\mathbf{d})$, $\mathcal{F}$ being a generic encoding function.

The evaluation of $\mathbf{c}$ depends on a generator polynomial $g(z) = \sum_{i=0}^{\ell(\mathbf{c})} a_i z^i$ characterizing the CRC [5]. A systematic generator matrix $\mathbf{G} = [\mathbf{I}, \mathbf{\Pi}]$ may be associated to $g(z)$. Using $\mathbf{G}$, $\mathbf{c}$ may be determined by a recursive processing over the $\ell(\mathbf{d})$ bits of $\mathbf{d}$ as follows

$$\mathbf{c}^{j+1} = \mathcal{F}(\mathbf{d}^{j+1}) = \mathbf{c}^j \oplus (d_{j+1} \cdot \boldsymbol{\pi}(d_{j+1})). \tag{2}$$

In (2), $\mathbf{d}^j = [d_1 \dots d_j, 0 \dots 0]$, $\boldsymbol{\pi}(d_j)$ is the j-th row of $\mathbf{\Pi}$, *i.e.*, the parity vector related to d_j, and $\oplus$ represents the XOR operator. At initialization, $\mathbf{c}^0$ is set to $\mathbf{0}$. After $\ell(\mathbf{d})$ iterations, the vector $\mathbf{c}^{\ell(\mathbf{d})}$ contains the CRC value related to $\mathbf{d}$ (*i.e.*, $\mathbf{c}^{\ell(\mathbf{d})} = \mathbf{c}$).

The vector $\mathbf{t}$ is then BPSK-modulated and transmitted over an AWGN channel that corrupts the modulated packets with a Gaussian noise of zero mean and variance σ^2. At the receiver, the observed vector is $\mathbf{y}_t = [\mathbf{y}_h, \mathbf{y}_x, \mathbf{y}_c]$, where $\mathbf{y}_h$, $\mathbf{y}_x$, and $\mathbf{y}_c$ are the observations of $\mathbf{h}$, $\mathbf{x}$, and $\mathbf{c}$ respectively. $\mathbf{y}_t$ contains the observations of $\mathbf{t}$ and represents a segment of the APL packet depicted in Figure 2. An overview of the transmission scheme is illustrated in Figure 3.

In practice, $\mathbf{x}$ may be organized in groups of codewords (*e.g.*, texture information of a block or a macroblock), which are assumed to be encoded independently. Let $\mathbf{a}_1 \dots \mathbf{a}_E$ be the E groups of codewords composing $\mathbf{x}$, *i.e.*, $\mathbf{x} = [\mathbf{a}_1 \dots \mathbf{a}_E]$. The lengths $\ell(\mathbf{a}_e)$, for $e = 1 \dots E$, are supposed to be transmitted reliably as side information to the decoder. The decoding of a group may be performed by selecting the corresponding portion in the received packet.

B. Decoding algorithm

Assuming that the header $\mathbf{h}$ has been correcly received, the optimal MAP estimator $\widehat{\mathbf{a}}_e$ for the e-th group is given by

$$\widehat{\mathbf{a}}_e = \arg \max_{\mathbf{a}_e \in \Omega_a^e} P(\mathbf{a}_e | \mathbf{h}, \mathbf{y}_x, \mathbf{y}_c), \tag{3}$$

where Ω_a^e is the set of *valid* combinations of $\mathbf{a}_e$, *i.e.*, compliant with the syntax of the VLC and the semantic of the source. Nevertheless, since Ω_a^e is not well structured, obtaining $\widehat{\mathbf{a}}_e$ would require constructing the $2^{\ell(\mathbf{a}_e)}$ possible combinations, keeping only the valid sequences (belonging to Ω_a^e), then evaluating $P(\mathbf{a}_e|\mathbf{h}, \mathbf{y}_x, \mathbf{y}_c)$ for each of them. When $\ell(\mathbf{a}_e)$ is large (which is usually the case to reduce the overhead due to the transmission of side information), a sequential decoder is involved in order to reduce the decoding complexity [1].

Considering the n-th step of the decoding of group e. One may write

$$\mathbf{x} = [\mathbf{b}_e, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{r}_{e,n}],$$

with :

- $\mathbf{b}_e = [\mathbf{a}_1 \dots \mathbf{a}_{e-1}]$, the bits of the first $e-1$ groups. Note that for the decoding of $\mathbf{a}_e$, $\mathbf{b}_e$ is considered as a random vector and not as the decoded bitstream obtained previously.
- $\mathbf{u}_{e,n}$, the first bits of $\mathbf{a}_e$ for which a set of valid combinations $\Omega_u^{e,n}$ has been evaluated at step $n-1$ by the decoder.
- $\mathbf{s}_{e,n}$, a vector for which, regardless of the syntax of the VLC and the semantic of the video coder, $2^{\ell(\mathbf{s}_{e,n})}$ binary combinations are possible. Let $\Omega_s^{e,n}$ be the set of these sequences.
- $\mathbf{r}_{e,n}$, the $\ell(\mathbf{r}_{e,n})$ remaining bits of $\mathbf{x}$. These bits have been not yet processed by the decoder but they do influence the CRC.

Figure 4 illustrates the considered partition of the packet. The observations associated to these four vectors are $\mathbf{y}_b^e$, $\mathbf{y}_u^{e,n}$, $\mathbf{y}_s^{e,n}$, and $\mathbf{y}_r^{e,n}$. Moreover, let $\Omega_{[u,s]}^{e,n} \subset \Omega_u^{e,n} \times \Omega_s^{e,n}$ be the set of valid pairs $[\mathbf{u}_{e,n}, \mathbf{s}_{e,n}]$.

At the n-th step, the sequential decoding algorithm evaluates

$$P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n} | \mathbf{y}_b^{e,n}, \mathbf{y}_u^{e,n}, \mathbf{y}_s^{e,n}, \mathbf{y}_r^{e,n}, \mathbf{h}) \propto P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{y}_b^{e,n}, \mathbf{y}_u^{e,n}, \mathbf{y}_s^{e,n}, \mathbf{y}_r^{e,n} | \mathbf{h}). \tag{4}$$

for each $[\mathbf{u}_{e,n}, \mathbf{s}_{e,n}] \in \Omega_u^{e,n} \times \Omega_s^{e,n}$. In (4), one may write

$$P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{y}_b^{e,n}, \mathbf{y}_u^{e,n}, \mathbf{y}_s^{e,n}, \mathbf{y}_r^{e,n} | \mathbf{h}) = \sum_{\mathbf{b}_e} \sum_{\mathbf{r}_{e,n}} P(\mathbf{b}_e, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{r}_{e,n}, \mathbf{y}_b^{e,n}, \mathbf{y}_u^{e,n}, \mathbf{y}_s^{e,n}, \mathbf{y}_r^{e,n} | \mathbf{h}). \tag{5}$$

Moreover

$$P(\mathbf{b}_e, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{r}_{e,n}, \mathbf{y}_b^{e,n}, \mathbf{y}_u^{e,n}, \mathbf{y}_s^{e,n}, \mathbf{y}_r^{e,n}|\mathbf{h}) = P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n}|\mathbf{h})P(\mathbf{y}_u^{e,n}|\mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{h}) \\ P(\mathbf{y}_s^{e,n}|\mathbf{y}_u^{e,n}, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{h})P(\mathbf{b}_e, \mathbf{r}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_r^{e,n}, \mathbf{y}_c|\mathbf{y}_u^{e,n}, \mathbf{y}_s^{e,n}, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{h}). \tag{6}$$

Using the fact that $\mathbf{u}_{e,n}$ and $\mathbf{s}_{e,n}$ do not depend on $\mathbf{h}$ and that the channel is memoryless, (6) becomes

$$P(\mathbf{b}_e, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{r}_{e,n}, \mathbf{y}_b^{e,n}, \mathbf{y}_u^{e,n}, \mathbf{y}_s^{e,n}, \mathbf{y}_r^{e,n}|\mathbf{h}) \\ = P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n})P(\mathbf{y}_u^{e,n}|\mathbf{u}_{e,n})P(\mathbf{y}_s^{e,n}|\mathbf{s}_{e,n})P(\mathbf{b}_e, \mathbf{r}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_r^{e,n}, \mathbf{y}_c|\mathbf{h}, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}). \tag{7}$$

Now, combining (4), (5), and (7), one obtains

$$P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n}|\mathbf{y}_b^{e,n}, \mathbf{y}_u^{e,n}, \mathbf{y}_s^{e,n}, \mathbf{y}_r^{e,n}, \mathbf{h}) \\ \propto P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n})P(\mathbf{y}_u^{e,n}|\mathbf{u}_{e,n})P(\mathbf{y}_s^{e,n}|\mathbf{s}_{e,n})\Phi(\mathbf{h}, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_r^{e,n}, \mathbf{y}_c), \tag{8}$$

with

$$\Phi(\mathbf{h}, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_r^{e,n}, \mathbf{y}_c) = \sum_{\mathbf{b}_e, \mathbf{r}_{e,n}} P(\mathbf{b}_e, \mathbf{r}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_r^{e,n}, \mathbf{y}_c|\mathbf{h}, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}). \tag{9}$$

In (8), $P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n})$ represents the *a priori* probability of sequence $[\mathbf{u}_{e,n}, \mathbf{s}_{e,n}]$, which is null if $[\mathbf{u}_{e,n}, \mathbf{s}_{e,n}] \notin \Omega_{[u,s]}^{e,n}$. As for the valid sequences, they are assumed to be equally likely *a priori*, *i.e.*, $P(\mathbf{u}_{e,n}, \mathbf{s}_{e,n}) = 1/|\Omega_{[u,s]}^{e,n}|$. Consequently, the metric $\mathcal{M}_e$ associated to a *valid* sequence in group e is given by

$$\mathcal{M}_e([\mathbf{u}_{e,n}, \mathbf{s}_{e,n}] \in \Omega_{[u,s]}^{e,n}|\mathbf{h}, \mathbf{y}_t) = P(\mathbf{y}_u^{e,n}|\mathbf{u}_{e,n})P(\mathbf{y}_s^{e,n}|\mathbf{s}_{e,n})\Phi(\mathbf{h}, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_r^{e,n}, \mathbf{y}_c), \tag{10}$$

where $P(\mathbf{y}_u^{e,n}|\mathbf{u}_{e,n})$ and $P(\mathbf{y}_s^{e,n}|\mathbf{s}_{e,n})$ are the likelihoods of $\mathbf{u}_{e,n}$ and $\mathbf{s}_{e,n}$ respectively.

C. Implementation issues and complexity

In (10), $\Phi(\mathbf{h}, \mathbf{u}_{e,n}, \mathbf{s}_{e,n}, \mathbf{y}_b^e, \mathbf{y}_r^{e,n}, \mathbf{y}_c)$ is a sum the complexity of which is $\mathcal{O}(2^{\ell(\mathbf{b}_e)+\ell(\mathbf{r}_{e,n})})$. Consequently, the evaluation complexity of (4) for all $[\mathbf{u}_{e,n}, \mathbf{s}_{e,n}] \in \Omega_{[u,s]}^{e,n}$ is $\mathcal{O}(|\Omega_u^{e,n}| \cdot |\Omega_s^{e,n}| \cdot 2^{\ell(\mathbf{b}_e)+\ell(\mathbf{r}_{e,n})})$. $|\Omega_s^{e,n}|$ depends on the number of bits taken into account at the n-th steps and may thus be upper bounded by a constant. The main difficulty comes from $|\Omega_u^{e,n}|$, which is growing exponentially with n. To limit the complexity increase, at each step, only the M most probable sequences belonging to $\Omega_{[u,s]}^{e,n}$ are kept and stored in $\Omega_u^{e,n+1}$. The parameter M allows to tune the trade-off between complexity and efficiency. A each step, one obtains a suboptimal algorithm the complexity of which becomes $\mathcal{O}(2^{\ell(\mathbf{b}_e)+\ell(\mathbf{r}_{e,n})})$, mainly due to the evaluation of Φ in (10).

Section IV describes optimal and suboptimal reduced-complexity algorithms for determining Φ and $\mathcal{M}_e$.

Let N_e be the number of steps necessary to reach the end of group e. The number of bits $\ell(\mathbf{s}_{e,n})$, for $i = 1 \dots N_e$, must thus be adjusted such that

$$\sum_{i=1}^{N_e} \ell(\mathbf{s}_{e,i}) = \ell(\mathbf{a}_e), \tag{11}$$

for all $e = 1 \dots E$. In practice, the first $N_e - 1$ decoding depths are set to a constant value and the last one, *i.e.*, $\ell(\mathbf{s}_{e,N_e})$, is chosen so that (11) is satisfied.

We now describe the complete sequential decoding algorithm for the e-th group. At initialization ($n = 1$), $\Omega_u^{e,1} = \emptyset$. Afterwards, at each step $n > 1$, the algorithm explores the new branches (on $\ell(\mathbf{s}_{e,n})$ bit depth) and only preserves the M most probable extended sequences $[\mathbf{u}_{e,n}, \mathbf{s}_{e,n}]$. These M sequences are temporarily stored in a stack (corresponding to $\Omega_u^{e,n+1}$), before being extended again at the next step. Figure 5 illustrates the evolution of parts $\mathbf{b}_e$, $\mathbf{u}_{e,n}$, $\mathbf{s}_{e,n}$, and $\mathbf{r}_{e,n}$ through the different steps. The flowchart of the decoding algorithm is depicted in Figure 6. Note that the metric $\mathcal{M}([\mathbf{u}_{e,n}, \mathbf{s}_{e,n}]|\mathbf{h}, \mathbf{y}_t)$ is computed using (10).

In Section V, this algorithm is applied to the decoding of H.264/AVC CAVLC sequences.

IV. Practical Evaluation of the MAP Metric

For the sake of simplicity, the exponents e and n are omitted in what follows. Moreover, $\Phi(\mathbf{h}, \mathbf{u}, \mathbf{s}, \mathbf{y}_b, \mathbf{y}_r, \mathbf{y}_c)$ and $\mathcal{M}([\mathbf{u}, \mathbf{s}] \in \Omega_{[u,s]}|\mathbf{h}, \mathbf{y}_t)$ are replaced by Φ and $\mathcal{M}([\mathbf{u}, \mathbf{s}])$.

In (10), only Φ is complex to evaluate. Assuming that the bits of $\mathbf{b}$ and $\mathbf{r}$ are i.i.d. and do not depend on $\mathbf{h}$, $\mathbf{u}$, and $\mathbf{s}$, (9) becomes

$$\Phi = \sum_{\mathbf{b}} \sum_{\mathbf{r}} P(\mathbf{b}) P(\mathbf{y}_b|\mathbf{b}) P(\mathbf{r}) P(\mathbf{y}_r|\mathbf{r}) P(\mathbf{y}_c|\mathcal{F}([\mathbf{h}, \mathbf{b}, \mathbf{u}, \mathbf{s}, \mathbf{r}])). \tag{12}$$

Assuming that all $\mathbf{b}$ and all $\mathbf{r}$ are equally likely *a priori*, the evaluation of (12) requires summing the product of the likelihoods related to $\mathbf{b}$, $\mathbf{r}$, and their corresponding CRC, over the $2^{\ell(\mathbf{b})+\ell(\mathbf{r})}$ combinations of $\mathbf{b}$ and $\mathbf{r}$. In this section, two reduced-complexity methods are proposed for evaluating (10) by optimizing the evaluation of Φ. The first provides an exact evaluation of $\mathcal{M}$, whereas the second results in an approximate evaluation of the metric.

A. Exact Computation

The CRC may be evaluated recursively over the data $\mathbf{d}$, as shown by (2). More precisely, the value of the CRC associated to the first $j+1$ bits of $\mathbf{d}$ (shortly, at time $j+1$) only depends on the value of the CRC at time j and on the $j+1$-st bit of $\mathbf{d}$. Each value of the CRC at time j leads to two different values of the CRC at time $j+1$. Consequently, the evolution of the CRC values according to the bits of $\mathbf{d}$ can be described by a trellis. In this trellis, states correspond to the $2^{\ell(\mathbf{c})}$ possible values of the CRC. Transitions are determined by the bits of $\mathbf{d}$. At each time $j = 1 \dots \ell(\mathbf{d})$, we study the contribution of d_j (the j-th bit of $\mathbf{d}$) over the global CRC.

In our case, $\mathbf{d} = [\mathbf{h}, \mathbf{b}, \mathbf{u}, \mathbf{s}, \mathbf{r}]$. The header $\mathbf{h}$ is assumed to be known and we want to find the best combination of $[\mathbf{u}, \mathbf{s}] \in \Omega_{[u,s]}$ by taking into account the redundancy of the code (given by $\mathbf{c}$). The trellis is thus applied to the portions $\mathbf{b}$, $\mathbf{r}$, and $\mathbf{c}$ for given $\mathbf{h}$, $\mathbf{u}$, and $\mathbf{s}$. This trellis consists in grouping combinations of $\mathbf{b}$ and $\mathbf{r}$ giving the same value of CRC.

Consequently, (12) may be rewritten as

$$\Phi = \sum_{\mathbf{c}} P(\mathbf{y}_c|\mathbf{c}) \sum_{\mathbf{b},\mathbf{r}|\mathcal{F}([\mathbf{h},\mathbf{b},\mathbf{u},\mathbf{s},\mathbf{r}])=\mathbf{c}} P(\mathbf{b})P(\mathbf{y}_b|\mathbf{b})P(\mathbf{r})P(\mathbf{y}_r|\mathbf{r}). \tag{13}$$

In the sequel, the state associated to a possible value $\mathbf{c}'$ of CRC is denoted by $S(\mathbf{c}')$, $\mathbf{c}'$ being the binary representation of $S(\mathbf{c}') \in \{0 \dots 2^{\ell(\mathbf{c})} - 1\}$. For instance with a 3-bit CRC, if $\mathbf{c}' = [1, 0, 1]$ then $S(\mathbf{c}') = 5$. After some derivations, one can show that (13) may be generalized as follows (see Appendix)

$$\begin{aligned} \Phi &= \sum_{\mathbf{c}'} \left[\sum_{\mathbf{b}|\mathcal{F}([\mathbf{h},\mathbf{b},\mathbf{0},\mathbf{0},\mathbf{0}])=\mathbf{c}'} P(\mathbf{b})P(\mathbf{y}_b|\mathbf{b}) \right] \left[\sum_{\mathbf{r}} P(\mathbf{r})P(\mathbf{y}_r|\mathbf{r})P(\mathbf{y}_c|\mathbf{c}' \oplus \mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{r}])) \right] \\ &= \sum_{\mathbf{c}'} \alpha(S(\mathbf{c}')) \cdot \beta(S(\mathbf{c}' \oplus \mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{0}]))), \end{aligned} \tag{14}$$

with

$$\alpha(S(\mathbf{c}')) = \sum_{\mathbf{b}|\mathcal{F}([\mathbf{h},\mathbf{b},\mathbf{0},\mathbf{0},\mathbf{0}])=\mathbf{c}'} P(\mathbf{b})P(\mathbf{y}_b|\mathbf{b}), \tag{15}$$

$$\beta(S(\mathbf{c}'')) = \sum_{\mathbf{r}} P(\mathbf{r})P(\mathbf{y}_r|\mathbf{r})P(\mathbf{y}_c|\mathbf{c}'' \oplus \mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{r}])), \tag{16}$$

for all $\mathbf{c}', \mathbf{c}'' \in GF(2)^{\ell(\mathbf{c})}$. In (15), $\alpha(S(\mathbf{c}'))$ represents the sum of the probabilities associated to the combinations of $\mathbf{b}$ reaching state $S(\mathbf{c}')$ when starting from state $S(\mathcal{F}([\mathbf{h},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}]))$. In (16), $\beta(S(\mathbf{c}''))$ denotes the sum of the probabilities associated to all combinations of $[\mathbf{r}, \mathbf{c}'' \oplus$

$\mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{r}])]$ when starting from state $S(\mathbf{c}'')$. In fact, the evaluation of Φ using (14) is efficiently performed using the BCJR algorithm for block codes [2], [29]. Thus, $\alpha(S(\mathbf{c}'))$ and $\beta(S(\mathbf{c}''))$ are easily evaluated recursively as follows

$$\begin{aligned}\alpha_{j+1}(S(\mathbf{c}')) &= P(b_{j+1}=0)P(y_{b_{j+1}}|b_{j+1}=0)\alpha_j(S(\mathbf{c}')) \\ &+ P(b_{j+1}=1)P(y_{b_{j+1}}|b_{j+1}=1)\alpha_j(S(\mathbf{c}'\oplus\boldsymbol{\pi}(b_{j+1}))),\end{aligned} \tag{17}$$

with the boundary conditions (at $j=0$)

$$\alpha_0(S(\mathbf{c}')) = \begin{cases} 1 & \text{for } \mathbf{c}' = \mathcal{F}([\mathbf{h},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}]) \\ 0 & \text{for all } \mathbf{c}' \neq \mathcal{F}([\mathbf{h},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}]) \end{cases}, \tag{18}$$

and

$$\begin{aligned}\beta_{j-1}(S(\mathbf{c}'')) &= P(r_j=0)P(y_{r_j}|r_j=0)\beta_j(S(\mathbf{c}'')) \\ &+ P(r_j=1)P(y_{r_j}|r_j=1)\beta_j(S(\mathbf{c}''\oplus\boldsymbol{\pi}(r_j))),\end{aligned} \tag{19}$$

with the boundary conditions (at $j=\ell(\mathbf{r})$)

$$\beta_{\ell(\mathbf{r})}(S(\mathbf{c}'')) = P(\mathbf{y}_c|\mathbf{c}''), \text{ for all } \mathbf{c}''\in GF(2)^{\ell(\mathbf{c})}. \tag{20}$$

The equations in (17) and (19) are the key for computing $\alpha(S(\mathbf{c}'))$ with a forward recursion over the bits of $\mathbf{b}$ and $\beta(S(\mathbf{c}''))$ with a backward recursion over the bits of $\mathbf{r}$. After $\ell(\mathbf{b})$ iterations, $\alpha_{\ell(\mathbf{b})}(S(\mathbf{c}')) = \alpha(S(\mathbf{c}'))$, and after $\ell(\mathbf{r})$ iterations, $\beta_0(S(\mathbf{c}'')) = \beta(S(\mathbf{c}''))$.

Finally, substituting (14) in (10), one obtains

$$\begin{aligned}\mathcal{M}([\mathbf{u},\mathbf{s}]) &= \sum_{\mathbf{c}'} \alpha(S(\mathbf{c}'))\cdot P(\mathbf{y}_u|\mathbf{u})P(\mathbf{y}_s|\mathbf{s})\cdot\beta(S(\mathbf{c}'\oplus\mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{0}])))\\ &= P(\mathbf{y}_u|\mathbf{u})P(\mathbf{y}_s|\mathbf{s}) \sum_{\mathbf{c}',\mathbf{c}''|\mathbf{c}''=\mathbf{c}'\oplus\mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{0}])} \alpha(S(\mathbf{c}'))\cdot\beta(S(\mathbf{c}'')).\end{aligned} \tag{21}$$

The evaluation of $\mathcal{M}([\mathbf{u},\mathbf{s}])$ consists in summing the probabilities associated to the $2^{\ell(\mathbf{c})}$ paths linking state $S(\mathbf{c}')$ to state $S(\mathbf{c}'')$, such as $\mathbf{c}''=\mathbf{c}'\oplus\mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{0}])$.

The steps for evaluating the global metric (10) with the above mentioned method are summarized below:

Step 1: Initialize $\alpha_0(S(\mathbf{c}'))$ and $\beta_{\ell(\mathbf{r})}(S(\mathbf{c}''))$ according to (18) and (20).

Step 2: Compute $\alpha_j(S(\mathbf{c}'))$, for all $\mathbf{c}'\in GF(2)^{\ell(\mathbf{c})}$ and for all $j=1\ldots\ell(\mathbf{b})$, by using (17) in a forward way (partial BCJR forward step).

Step 3: Compute $\beta_j(S(\mathbf{c}''))$, for all $\mathbf{c}''\in GF(2)^{\ell(\mathbf{c})}$ and for all $j=\ell(\mathbf{r})-1\ldots 0$, by using (19) in a backward way (partial BCJR backward step).

Step 4: For each $[\mathbf{u},\mathbf{s}] \in \Omega_{[u,s]}$, compute the metric $\mathcal{M}([\mathbf{u},\mathbf{s}])$ by using (21), recalling that $\alpha(S(\mathbf{c}')) = \alpha_{\ell(\mathbf{b})}(S(\mathbf{c}'))$ and $\beta(S(\mathbf{c}'')) = \beta_0(S(\mathbf{c}''))$.

Hence, one step of the sequential decoding is performed with a complexity $\mathcal{O}((\ell(\mathbf{b})+\ell(\mathbf{r})+|\Omega_{[u,s]}|)2^{\ell(\mathbf{c})})$, compared to $\mathcal{O}(|\Omega_{[u,s]}|2^{\ell(\mathbf{b})+\ell(\mathbf{r})})$ for a decoding with a straightforward metric computation.

Remark 1: As presented above, the decoding of $\mathbf{x}$ requires repeating steps 1 to 4 for each portion $[\mathbf{u},\mathbf{s}]$ in $\mathbf{x}$ since the portions $\mathbf{b}$ and $\mathbf{r}$ change according to the position of $[\mathbf{u},\mathbf{s}]$. To optimize the global decoding, as soon as $\mathbf{y}_t$ is received, we can compute each value of $\alpha_j(S(\mathbf{c}'))$ and $\beta_j(S(\mathbf{c}''))$, for all $\mathbf{c}',\mathbf{c}'' \in GF(2)^{\ell(\mathbf{c})}$ and for all $j = 0 \dots \ell(\mathbf{x})$, and store them in matrices $\mathbf{A}$ and $\mathbf{B}$. This is equivalent to perform a *complete* BCJR algorithm over $\mathbf{x}$: the forward step is performed on $\mathbf{b} = \mathbf{x}$ and the backward step on $\mathbf{r} = \mathbf{x}$. The global decoding of $\mathbf{x}$ begins after this step. As explained previously, each portion $[\mathbf{u},\mathbf{s}]$ is sequentially decoded by using (21) in which the values of $\alpha(S(\mathbf{c}'))$ and $\beta(S(\mathbf{c}''))$ are extracted from $\mathbf{A}$ and $\mathbf{B}$ depending on the position of the current portion $[\mathbf{u},\mathbf{s}]$.

Note that in this case, steps 1 to 3 are performed once as a preamble, and step 4 is performed repeatedly for each $[\mathbf{u},\mathbf{s}]$ in $\mathbf{x}$.

B. Approximate Computation

In practice, most CRCs are larger than 16 bits and the complexity $\mathcal{O}(2^{\ell(\mathbf{c})})$ is too large to allow a real-time implementation of the method presented in Section IV-A. An approximate computation consists in splitting the CRC into m_b partitions of $\ell(\mathbf{c})/m_b$ bits, each partition being assumed statistically independent from the others. A trellis may be associated to each of the m_b partitions. Thus, $\mathbf{y}_c$ may be written as $\mathbf{y}_c = [\mathbf{y}_{c_1} \dots \mathbf{y}_{c_{m_b}}]$. Using the independence assumption, as explained with more details in [17], the global metric in (21) becomes

$$\mathcal{M}([\mathbf{u},\mathbf{s}]) = P(\mathbf{y}_u|\mathbf{u})P(\mathbf{y}_s|\mathbf{s}) \prod_{m=1}^{m_b} \sum_{\mathbf{c}'_m,\mathbf{c}''_m|\mathbf{c}''_m=\mathbf{c}'_m\oplus\mathcal{F}_m([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{0}])} \alpha^m(S(\mathbf{c}'_m)) \cdot \beta^m(S(\mathbf{c}''_m)), \quad (22)$$

where $\alpha^m(S(\mathbf{c}'_m))$ and $\beta^m(S(\mathbf{c}''_m))$ represent the probabilities associated to states $S(\mathbf{c}'_m)$ and $S(\mathbf{c}''_m)$ respectively, for $\mathbf{c}'_m,\mathbf{c}''_m \in GF(2)^{\ell(\mathbf{c})/m_b}$, in the m-th trellis.

The total complexity for evaluating (22) is now $\mathcal{O}((\ell(\mathbf{b})+\ell(\mathbf{r})+|\Omega_{[u,s]}|)m_b 2^{\ell(\mathbf{c})/m_b})$, at the cost of a slightly suboptimal performance.

Remark 2: To reduce the complexity of the global decoding of $\mathbf{x}$, we can apply the principle introduced in Remark 1 to the new method. In this case, the algorithm generates, as a preamble, the m_b submatrices $\mathbf{A}^m$ and $\mathbf{B}^m$ associated to partition $\mathbf{c}_m$. During the decoding, the values of $\alpha^m(S(\mathbf{c}'_m))$ and $\beta^m(S(\mathbf{c}''_m))$ in (22) are extracted from $\mathbf{A}^m$ and $\mathbf{B}^m$ according to the position of the current portion $[\mathbf{u}, \mathbf{s}]$.

V. Simulation Results

In the *extended* profile of H.264/AVC [10], an error-resilience mode is provided. In this mode, the compressed picture data are classified according to their influence on the video quality. Three partitions are defined:

- Partition A contains the headers and the motion vectors of each encoded picture.
- Partition B consists of the texture coefficients of the INTRA coded blocks.
- Partition C regroups the texture coefficients of INTER coded blocks.

This stream decomposition allows an adjustment of the protection to the sensitivity of the partition to be sent. After compression, each partition is encapsulated in a NALU (*Network Abstraction Layer Unit*) which is delivered to the RTP layer. Packets associated to the A partition are assumed heavily protected and correctly interpreted at the receiver. On the other hand, B and C packets are transmitted over a noisy channel and are corrupted by transmission errors. As previously mentioned, these packets contain the texture coefficients of the different 4×4 blocks of a picture. These blocks are encoded in CAVLC [22].

In this paper, we focus on the decoding of the CAVLC sequences included in the B and C packets. Each CAVLC sequence is considered as an independent group of codewords which can be separated from the others by using synchronization markers, transmitted as side information. Consequently, the group-based sequential decoding method of Section III may be used for their estimation. Note that in H.264/AVC, the CAVLC sequences are not totally independent (adaptive context) but the small existing dependencies may be neglected. The performance of the presented method has been evaluated by simulations along with that of two other decoding methods: a standard decoding method and a classical robust decoding method (exploiting only the source properties).

The simulated system consists of a transmitter, a channel, and a receiver. The transmitter uses repeatedly the 5 first pictures of *Foreman.cif* with the IPPPP frame structure and generates the

encoded partitions using the CAVLC H.264/AVC video coder. Video packets (partitions) are then processed by the protocol stack defined in Figure 1. At the MAC layer, IP packets are fragmented in several MAC packets of variable payload size. A CRC of 4 bytes, consistent with the 802.11 standard, is added at the end of each MAC fragment. At the PHY layer of the transmitter, the data are encoded by the convolutional channel coder of the $802.11a$ standard. Next, the coded PHY packets are mapped onto BPSK symbols before being sent over the physical medium. To improve the decoding performance, the aforementioned position markers are sent as side information, indicating the location of each 4×4 encoded texture block in B and C packets. This side information is transmitted in a specific NALU and the markers are compressed using the Exp-Golomb coding of H.264/AVC. The overhead due to the transmission of this redundancy represents about 30 % of the total bitrate. The channel does not degrade the data contained in A packets nor the side information. On the other hand, it does add a white Gaussian noise, with a configurable variance, to the other packets. At the receiver, the data are processed by a SISO channel decoder and are then delivered to the APL layer (following the permeable mechanism explained in Section II). At the APL layer, as previously mentioned, three different decoders are considered:

1) A *standard* decoder performs hard decisions on the received soft data and makes usage of position markers to decode each block.
2) A *robust* decoder uses the source properties, the soft data as well as the position markers, but does not use the redundancy provided by the CRC. This decoder exploits the algorithm depicted in Section III, but the metric in (10) does not include the term Φ.
3) A *CRC-robust* decoder combines all the previous sources of redundancy along with the CRC properties through the decoding method presented in Section III.

Note that, in our simulations, the two robust decoders use the stack size $M = 20$ and the default decoding depth $\ell(\mathbf{s}) = 4$ bits. Also, the CRC-robust decoder uses the suboptimal method presented in Section IV-B. For this purpose, the CRC is split into 4 blocks of 8 bits.

Figures 8 and 7 plot the evolution of the PSNR (*Peak Signal to Noise Ratio*) of the decoded video as a function of the SNR for the three different decoders, with and without channel coding respectively. In Figure 7, the channel coding/decoding at PHY layer was deactivated. In both figures, the standard, robust, and CRC-robust decoders are compared for a MAC payload size of

100 bytes. We can notice that the standard decoder is outperformed by the two robust decoders in both cases. Moreover, the two robust decoders are equivalent for low SNRs and the CRC-robust decoder starts to outperform the classical robust decoder beyond a given threshold. Above this threshold, the coding gain increases with the SNR. This behavior is specific to channel decoding performance. In our simulations, the threshold is about 8.5 dB in Figure 7, and 1.8 dB in Figure 8.

Figure 9 illustrates the 5-th image of the used *Foreman.cif* video sequence, along with its reproductions obtained after this image is transmitted and decoded by the standard, robust and CRC-robust decoders respectively. In this case, the channel coding/decoding is considered. This result was obtained with a payload size of 100 bytes and at an SNR of 2.8 dB for which the PSNR of the standard, robust and CRC-robust decoders are of 29, 35 and 38 dB respectively (see Figure 8). Obviously, the image obtained with the standard decoder contains many artifacts and is of a very poor quality. On the other hand, the robust decoder strongly improves the quality even though some distortions are still visible. Finally, no visual difference may be noticed between the original image and the image obtained by the CRC-robust decoder.

VI. Conclusion

In this paper, we have presented a MAP estimator for robust video decoding. The decoder jointly exploits the inherent source coder information along with the MAC layer CRC redundancy. The implementation of this MAP estimator was shown to be a combination of a sequential decoding algorithm along with the BCJR algorithm. We applied this method for H.264/AVC decoding of CAVLC sequences. Simulation results show that the information carried by the CRC does improve the decoding efficiency. More precisely, joint use of CRC and source properties becomes interesting above a certain threshold. Moreover, the bitrate used for transmission of side information is quite high in the presented experiments. Our aim is to reduce this overhead by considering synchronization markers indicating, *e.g.*, the location of each macroblock of 16×16 pixels.

APPENDIX

Below, we detail the derivation of (14). Assuming that the bits of $\mathbf{b}$ and $\mathbf{r}$ are i.i.d. and do not depend on $\mathbf{h}$, $\mathbf{u}$, and $\mathbf{s}$, Φ in (9) may be developed as follows

$$\begin{aligned}\Phi &= \sum_{\mathbf{b},\mathbf{r}} P(\mathbf{b})P(\mathbf{y}_b|\mathbf{b})P(\mathbf{r})P(\mathbf{y}_r|\mathbf{r})P(\mathbf{y}_c|\mathcal{F}([\mathbf{h},\mathbf{b},\mathbf{u},\mathbf{s},\mathbf{r}])) \\ &= \sum_{\mathbf{r}} P(\mathbf{r})P(\mathbf{y}_r|\mathbf{r}) \sum_{\mathbf{b}} P(\mathbf{b})P(\mathbf{y}_b|\mathbf{b})P(\mathbf{y}_c|\mathcal{F}([\mathbf{h},\mathbf{b},\mathbf{u},\mathbf{s},\mathbf{r}]))\end{aligned} \tag{23}$$

In (23), the sum over $\mathbf{b}$ is a sum over all the possible values that $\mathbf{b}$ can take, each value corresponding to a path in the trellis. On the other hand, any possible path $\mathbf{b}$ ends up at a state $S(\mathbf{c}') \in \{0\ldots, 2^{\ell(\mathbf{c})}-1\}$ (*i.e.*, one of the $2^{\ell(\mathbf{c})}$ possible states). As a result, summing over all the possible paths $\mathbf{b}$ is equivalent to summing over all the paths $\mathbf{b}$ that end up at state 0, and all the paths that end up at state 1, ..., and all the paths that end up at state $2^{\ell(\mathbf{c})}-1$. Hence, (23) becomes

$$\begin{aligned}\Phi &= \sum_{\mathbf{r}} P(\mathbf{r})P(\mathbf{y}_r|\mathbf{r}) \sum_{\mathbf{c}'} \sum_{\mathbf{b}|\mathcal{F}([\mathbf{h},\mathbf{b},\mathbf{0},\mathbf{0},\mathbf{0}])=\mathbf{c}'} P(\mathbf{r})P(\mathbf{y}_r|\mathbf{r})P(\mathbf{y}_c|\mathbf{c}'\oplus\mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{r}])) \\ &= \sum_{\mathbf{c}'} \sum_{\mathbf{b}|\mathcal{F}([\mathbf{h},\mathbf{b},\mathbf{0},\mathbf{0},\mathbf{0}])=\mathbf{c}'} \sum_{\mathbf{r}} P(\mathbf{b})P(\mathbf{y}_b|\mathbf{b})P(\mathbf{r})P(\mathbf{y}_r|\mathbf{r})P(\mathbf{y}_c|\mathbf{c}'\oplus\mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{r}])) \\ &= \sum_{\mathbf{c}'} \left[\sum_{\mathbf{b}|\mathcal{F}([\mathbf{h},\mathbf{b},\mathbf{0},\mathbf{0},\mathbf{0}])=\mathbf{c}'} P(\mathbf{b})P(\mathbf{y}_b|\mathbf{b})\right] \left[\sum_{\mathbf{r}} P(\mathbf{r})P(\mathbf{y}_r|\mathbf{r})P(\mathbf{y}_c|\mathbf{c}'\oplus\mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{r}]))\right] \\ &= \sum_{\mathbf{c}'} \alpha(S(\mathbf{c}'))\cdot\beta(S(\mathbf{c}'\oplus\mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{u},\mathbf{s},\mathbf{0}]))),\end{aligned}$$

with

$$\begin{aligned}\alpha(S(\mathbf{c}')) &= \sum_{\mathbf{b}|\mathcal{F}([\mathbf{h},\mathbf{b},\mathbf{0},\mathbf{0},\mathbf{0}])=\mathbf{c}'} P(\mathbf{b})P(\mathbf{y}_b|\mathbf{b}), \\ \beta(S(\mathbf{c}'')) &= \sum_{\mathbf{r}} P(\mathbf{r})P(\mathbf{y}_r|\mathbf{r})P(\mathbf{y}_c|\mathbf{c}''\oplus\mathcal{F}([\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{r}])).\end{aligned}$$

REFERENCES

[1] J. B. Anderson and S. Mohan. *Source and Channel Coding: An Algorithmic Approach.* Kluwer, 1991.

[2] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. Info. Theory*, 20:284–287, 1974.

[3] R. Bauer and J. Hagenauer. On variable length codes for iterative source/channel decoding. In *Proc. of DCC*, pages 272–282, Snowbird, UT, 1998.

[4] C. Bergeron and C. Lamy-Bergot. Soft-input decoding of variable-length codes applied to the H.264 standard. In *Proc. of MSP*, pages 87–90, 29 Sept.-1 Oct. 2004.

[5] R. E. Blahut. *Theory and Practice of Error Control Codes.* Addison-Wesley, Reading, MA, 1984.

[6] V. Buttigieg and P. Farrell. A MAP decoding algorithm for variable-length error-correcting codes. In *Codes and Cyphers: Cryptography and Coding IV*, pages 103–119, Essex, England, 1995. The Inst. of Mathematics and its Appl.

[7] J. Hagenauer. Source-controlled channel decoding. *IEEE Trans. Com.*, 43(9):2449–2457, 1995.

[8] M.-C. Hong, H. Schwab, L. P. Kondi, and A. K. Katsaggelos. Error concealment algorithms for compressed video. *Signal Processing: Image Communication*, 14:473–492, 1999.

[9] IEEE. 802.11, part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Technical report, 1999.

[10] ITU-T and ISO/IEC JTC 1. Advanced video coding for generic audiovisual services. Technical report, ITU-T Rec. H.264, and ISO/IEC 14496-10 AVC, nov. 2003.

[11] S. Kaiser and M. Bystrom. Soft decoding of variable-length codes. In *Proc. of ICC*, volume 3, pages 1203–1207, New Orleans, 2000.

[12] W.-Y. Kung, C.-S. Kim, and C.-C. J. Kuo. Spatial and temporal error concealment techniques for video transmission over noisy channels. *IEEE Trans. Circuits and Systems for Video Technology*, 16:789–802, 2006.

[13] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison Wesley, Boston, third edition, 2005.

[14] C. Lamy and S. Merigeault. Procédé de correction d'une trame erronée par un récepteur. French patent no. 0206501, 2002.

[15] C. Lee, M. Kieffer, and P. Duhamel. Soft decoding of VLC encoded data for robust transmission of packetized video. In *Proc. of ICASSP*, pages 737–740, 2005.

[16] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, 2003.

[17] C. Marin, Y. Leprovost, M. Kieffer, and P. Duhamel. Robust MAC-lite and soft header recovery for packetized multimedia transmission. *IEEE Trans. Com.*, 2008. submitted.

[18] H. Nguyen and P. Duhamel. Iterative joint source-channel decoding of variable length encoded video sequences exploiting source semantics. In *Proc. of ICIP*, 2004.

[19] H. Nguyen, P. Duhamel, J. Brouet, and D. Rouffet. Robust VLC sequence decoding exploiting additional video stream properties with reduced complexity. In *Proc. of ICME*, pages 375–378, June 2004. Taipei, Taiwan.

[20] J. W. Nieto and W. N. Furman. Cyclic redundancy check (CRC) based error method and device. US Patent US 2007/0192667 A1, Aug. 16 2007.

[21] L. Perros-Meilhac and C. Lamy. Huffman tree based metric derivation for a low-complexity soft VLC decoding. In *Proc. of ICC*, 2002.

[22] I. Richardson. *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*. John Wiley and Sons, 2003.

[23] T. Richardson and U. Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008.

[24] G. Sabeva, S. Ben Jamaa, M. Kieffer, and P. Duhamel. Robust decoding of H.264 encoded video transmitted over wireless channels. In *Proc. of MSP*, pages 9–12, Victoria, Canada, 2006.

[25] K. Sayood. *Introduction to Data Compression, Second Edition*. Morgan Kaufmann, San Francisco, 2000.

[26] R. Thobaben and J. Kliewer. On iterative source-channel decoding for variable-length encoded markov sources using a bit-level trellis. In *Proc. of SPAWC*, Rome, 2003.

[27] R. Thobanen and J. Kliewer. Robust decoding of variable-length encoded markov sources using a three-dimensional trellis. *IEEE Com. Letters*, 7(7):320–322, 2003.

[28] T. Tillo, M. Grangetto, and G. Olmo. A flexible error resilient scheme for JPEG 2000. In *Proc. of MSP*, pages 295–298, 29 Sept.-1 Oct. 2004.

[29] J. K. Wolf. Efficient maximum-likelihood decoding of linear block codes using a trellis. *IEEE Trans. Info. Theory*, 24:76–80, 1978.

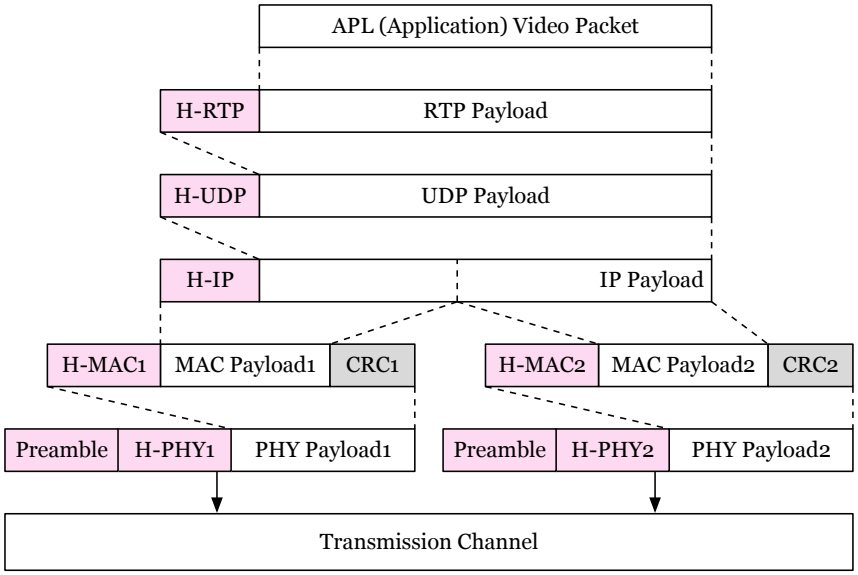


Fig. 1. Protocol stack for video transmission over WiFi

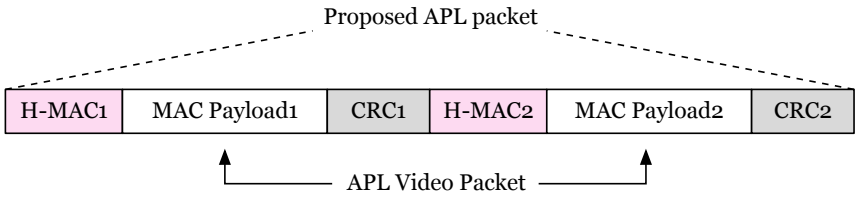


Fig. 2. Data available at the APL layer

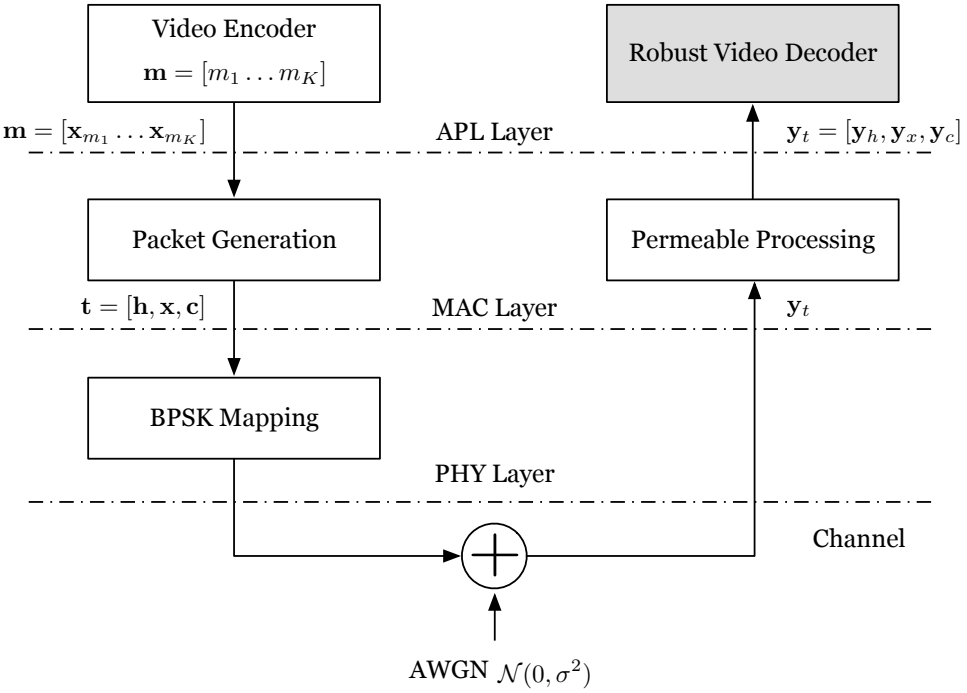


Fig. 3. Overview of the transmission scheme

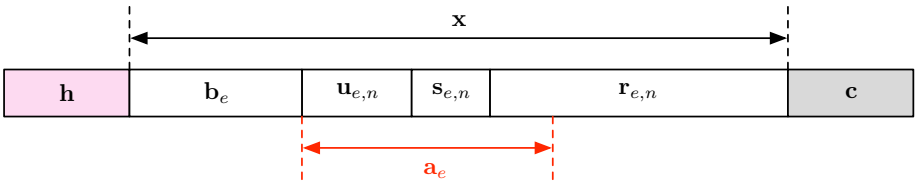


Fig. 4. Partitioning of the received packet

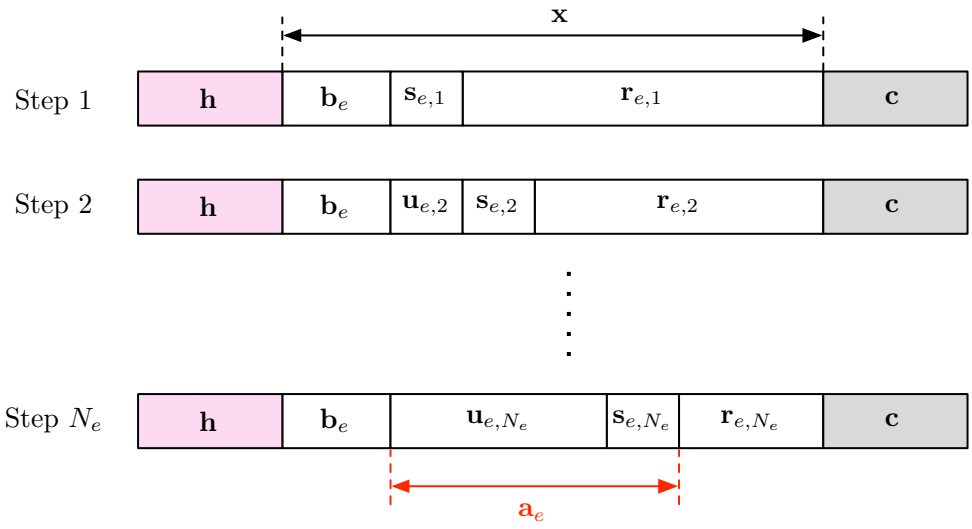


Fig. 5. Evolution of the partitions through the sequential decoding steps

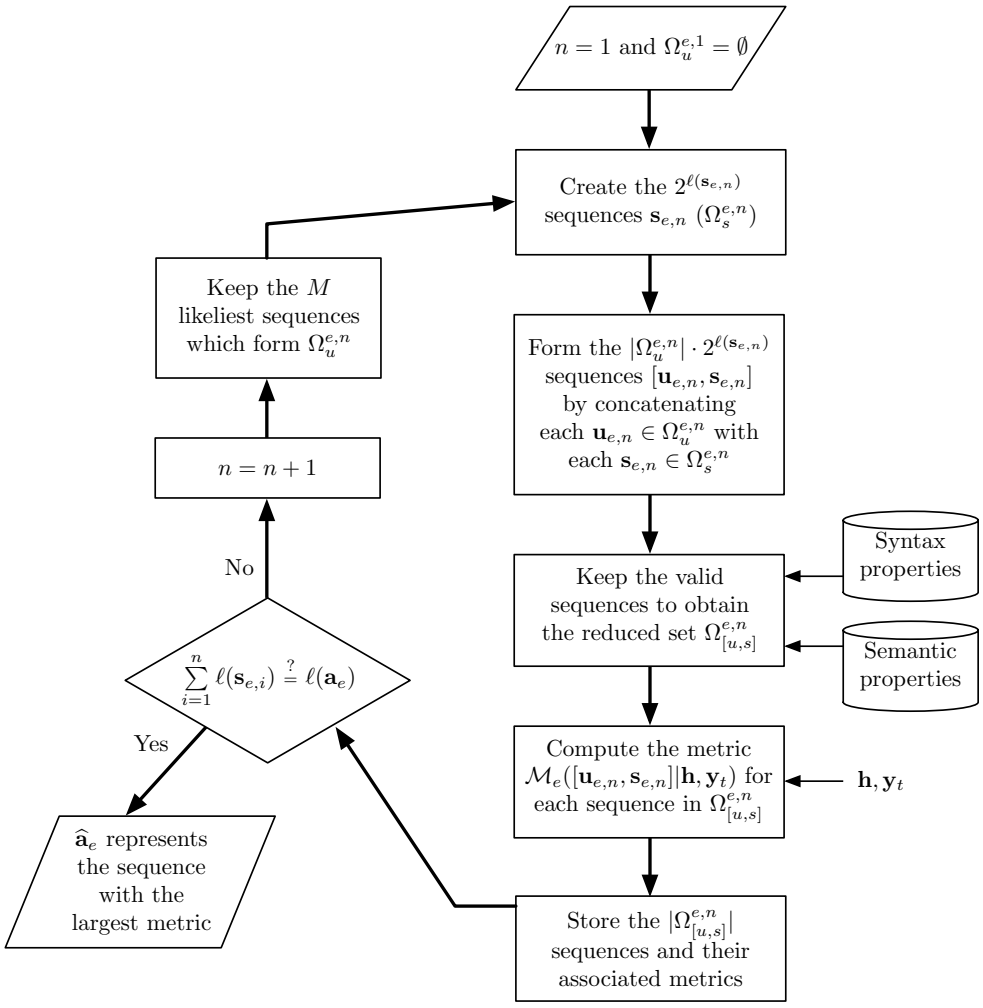


Fig. 6. Proposed sequential decoding scheme

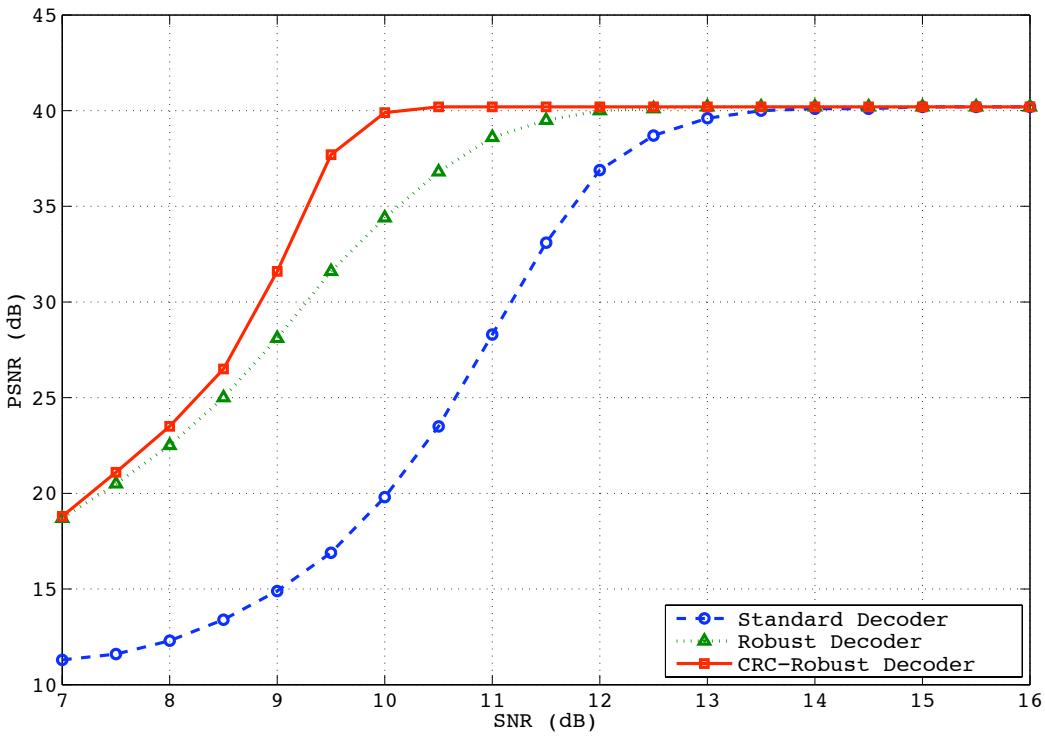


Fig. 7. PSNR vs SNR for a MAC payload size of 100 bytes, without channel coding

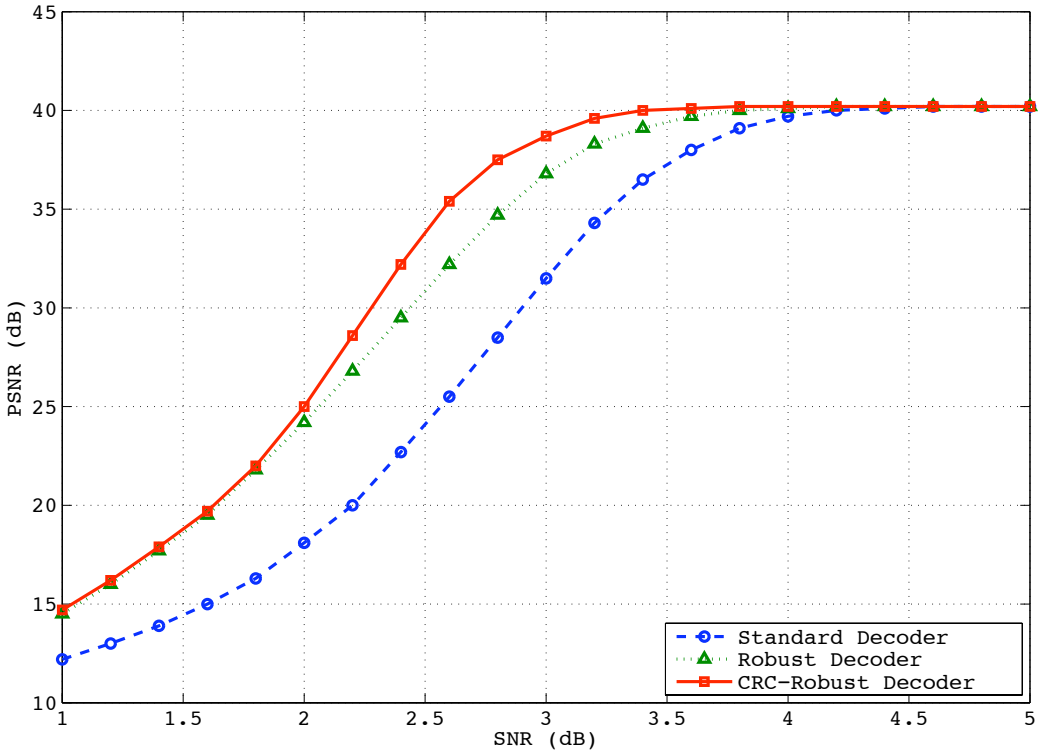


Fig. 8. PSNR vs SNR for a MAC payload size of 100 bytes, with channel coding

Fig. 9. Quality of the 5-th image of *Foreman.cif* obtained after (a) error-free decoding, (b) standard decoding, (c) robust decoding, and (d) CRC-robust decoding for a SNR of 2.8 dB and a MAC payload size of 100 bytes, with channel coding

Annexe C
Deuxième article IEEE

Robust MAC-Lite and Soft Header Recovery for Packetized Multimedia Transmission

Cédric Marin, Yann Leprovost, Michel Kieffer *Senior Member, IEEE*, and Pierre Duhamel *Fellow, IEEE*

Abstract

This paper presents an enhanced permeable layer mechanism useful for highly robust packetized multimedia transmission. Packet header recovery at various protocol layers using MAP estimation is the cornerstone of the proposed solution. The inherently available intra-layer and inter-layer header correlation proves to be very effective in selecting a reduced set of possible header configurations for further processing. The best candidate is then obtained through soft decoding of CRC protected data and CRC redundancy information itself. Simulation results for WiFi transmission using DBPSK modulated signals over AWGN channels show a substantial (4 to 12 dB) link budget improvement over classical hard decision procedures. We also introduce a sub-optimal and hardware realizable version of the proposed algorithm.

Index Terms

Codes, Communication systems, Decoding, MAP estimation, Protocols

I. INTRODUCTION

Due to bandwidth constraints, efficient transmission of multimedia contents requires the use of some source coding scheme [1]. Nevertheless, compressed data are very sensitive to transmission errors. A single corrupted bit may lead to a loss of a large amount of multimedia data at the receiver. Consequently, the bitstream entering the source decoder has to be almost error-free.

C. Marin and Y. Leprovost are with the Research and Innovation Center of Alcatel-Lucent, France.

M. Kieffer and P. Duhamel are with the L2S – CNRS - SUPELEC - Univ Paris-Sud, France.

This constraint is hardly satisfied when considering transmission over wireless channels. The data stream at receiver side may be heavily corrupted and not directly usable by the source decoder. A first solution to this problem consists in grouping data into packets protected by an error-detection code (CRC or checksum) [2], [3]. Packets, which have not been correctly received, are identified and can then be retransmitted. However, retransmissions may become difficult in scenarii with strong delay constraints, *e.g.*, for visiophony or may even become impossible when broadcasting data, *e.g.*, in satellite television.

In such situations, the standard solutions make use of very strong error-correcting codes (*e.g.*, turbo-codes, LDPC) at *Physical* (PHY) layer possibly combined with packet-erasure codes at intermediate protocol layers [4]. The redundancy introduced by these codes may however be oversized when the channel is good, reducing the bandwidth allocated for the data. In bad channel conditions, some corrupted packets still cannot be recovered and are assumed lost. Error-concealment techniques [5], [6] may then be used by the source decoders at *Application* (APL) layer. They exploit the redundancy (temporal and/or spatial) found in the multimedia data for reconstructing some information in place of the missing one.

In the recent years, joint source-channel decoding (JSCD) techniques have been proposed to correct damaged packets. These methods involve robust source decoders, which exploit the inherent redundancy in the received packets for correcting errors. Several sources of redundancy have been identified. Constraints in the syntax of variable-length source codes [7]–[9] have been used first. Redundancy due to the semantic of the source coders [10], [11] improve significantly the performance of robust decoders. Further redundancy due to the packetization of compressed data has been used in [12]. Altogether, the various redundancies can attain an unexpected amount. Furthermore, redundancy introduced by channel codes at physical layer can also be used in combination with residual redundancy to build iterative decoders as in [13]. These joint decoding schemes provide improved performance when compared to classical schemes, and could be of great use in many applications. However, they are not compliant with the standard protocol stacks in several ways: (i) they require exchange of soft information (*e.g.*, likelihood ratios) between the channel decoder at PHY layer and the robust source decoder at APL layer, (ii) they are not compatible with the use of acknowledgment procedures: a packet received in error needs not be retransmitted unless the robust receiver cannot recover the error, (iii) the headers of packets at a given layer must absolutely be available without error since they contain information necessary

for driving the layer in question (at the receiver).

Problem (i) above can be circumvented in some circumstances: a mobile receiver contains all the layers and can choose to forward soft values between layers. This paper assumes that it is the case. The main compatibility problem seems to be the third one: standard protocol stacks do not even allow damaged packets to reach the APL layer, the main reason being that the errors may impact some essential information contained in the headers, which is necessary even for the robust APL decoders.

This paper proposes some tools allowing to receive the various headers with an inherent robustness (even more than the robustness brought by JSCD to the payload) by using tools widely used in JSCD, and applied here to the higher protocol layers. More headers are thus correctly interpreted at each layer, increasing the number of packets reaching the APL layer. We show that robustness of the header is much higher than that of the corresponding payload, which is a prerequisite for implementing a fully *permeable* protocol layer mechanism [14]. In this paper, we implicitly assume that soft information is forwarded between layers.

The paper is organized as follows. After introducing the improved permeable layer mechanism in Section II, Section III derives the header recovery technique. Reduction of complexity is presented in Section IV. As an example, the design of the proposed mechanisms for PHY and MAC layers of WiFi is detailed in Section V. Finally, simulations are presented in Section VI.

II. Enhanced Permeable Layer Mechanism

Packetized multimedia transmission is usually based on an RTP/UDP/IP protocol stack [3]. Fig. 1 illustrates an example of segmentation and encapsulation mechanisms implemented at each protocol layer in case of a multimedia packet transmission with the 802.11 standard (WiFi) [15]. Error detection mechanisms implemented at each layer are detailed below.

At PHY layer, a known preamble allows the detection of the beginning of each PHY packet. A CRC protects the header fields (the preamble and the payload are not protected). Received packets with damaged headers are discarded. At MAC layer, a CRC protects the corresponding header and payload. When an error occurs, the packet is retransmitted. At IPv4 layer, the header fields are protected by a checksum. Received packets with damaged headers are discarded. At UDP layer, a checksum protects the header and the payload. When an error occurs, the packet is

discarded. We assume in this paper that packet fragmentation only occurs at MAC layer, which is a reasonable assumption for a wireless transmission.

The error-detection mechanisms provided by CRCs and checksums, combined with the retransmission mechanism at MAC layer, allow APL layer to receive only error-free packets. The price to be paid is a reduced throughput due to MAC level retransmissions which increase when the channel conditions worsen, or frequent use of error concealment when errors are detected at IPv4 or UDP layers (generally due to time-out constraint: the limit on the number of retransmissions at MAC level has been reached).

JSCD methods allow many errors to be corrected at APL layer based on soft information provided by lower protocol layers. The recently introduced UDP-Lite [16] mechanism, combined with lower permeable protocol layers [14], allow damaged APL packets to be fed to the APL layer. With UDP-Lite, a checksum protects a limited number of bytes (generally including the UDP-Lite, RTP, and APL header fields). Thus, packets with erroneous headers are still discarded. Considering the order of magnitude of the length of the packets and that of the various headers in actual wireless communications when tuned for difficult situations [17], this may happen more than expected. The bottleneck of such permeable transmission schemes is the fact that packets are discarded due to erroneous headers.

This paper proposes a method for recovering headers based on the various sources of redundancy in the protocol stack, thus increasing the amount of packets that can be used for robust decoding at APL layer. As a result, the efficiency of the decoding at APL layer is improved in all the cases : (i) when retransmissions are allowed, they are less frequent, (ii) when higher layer redundancy has been introduced to circumvent the problem (*e.g.*, the so-called MPE-FEC of the MAC layer in DVB-H), our strategy at least allows to reduce it, and finally, (iii) when no retransmission is allowed, it improves the quality of the multimedia content, because error concealment is used less frequently.

The proposed header recovery technique detailed in the next section involves two main ideas. First, intra-layer and inter-layer redundancy is present in the protocol stack. This redundancy has been exploited in the *Robust Header Compression* (ROHC) mechanism [18] by replacing the headers introduced by the RTP, UDP, and IP layers by a compressed version. Here, this redundancy is used to build some *a priori* information on the erroneous headers, improving their estimation. Second, CRCs and checksums are used as error-correcting codes, as proposed in [19]

and [20].

Header recovery combines soft information provided by the lower protocol layers, properties of the CRC or checksum, and *a priori* information obtained by a careful examination of the protocol. Fig. 2 illustrates how soft information is transmitted from layer $L-1$ to layer $L+1$ through the permeable layer L. This figure also summarizes the various sources of redundancy used to facilitate the header recovery at layer L. The payload at layer $L-1$ contains soft information on header, payload, and CRC at layer L. This header may itself be somewhat redundant with previous/future headers at layer L. In turn, the payload at layer L contains soft information on header, payload, and CRC at layer $L+1$. Albeit the header is removed after decoding, its information fields are necessary to help in the delivery of the payload to layer $L+1$, for further processing.

This paper focuses on the PHY and MAC layers of WiFi, as generic examples. In fact, the proposed permeable layer mechanism may be applied to any protocol layer.

III. MAP Estimator for Robust Header Recovery

In the sequel, $\ell(\mathbf{z})$ denotes the size of vector $\mathbf{z}$.

As a general situation, at a given layer L, the n-th incoming packet may include three items: a header, a payload, and a CRC. Information protected by the CRC $\mathbf{c}_n^L$ may have various properties, as far as the corresponding redundancy is concerned.

- The constant fields, represented by the vector $\mathbf{k}_n^L$, are assumed to be *known*.
- The *predictable* fields are embedded in the vector $\mathbf{p}_n^L$. In contrast with the known fields, the predictable fields are estimated by exploiting the *intra-layer* and *inter-layer redundancy* represented by R_n^L, which will be defined formally in what follows. They are predicted from information contained in the previously received packets. The predictable fields are assumed to be entirely determined if the previous packets have been correctly received.
- The important *unknown* fields are collected in the vector $\mathbf{u}_n^L$. These parameters are either completely unknown or limited to a configuration set $\Omega_u^L(\mathbf{k}_n^L, \mathbf{p}_n^L, R_n^L)$ the content of which is determined by the values of $\mathbf{k}_n^L$, $\mathbf{p}_n^L$, and R_n^L. This set contains the actual information on the data that the receiver must estimate.
- Finally, the vector $\mathbf{o}_n^L$ contains the *other* fields covered by the CRC. This last part contains unknown data, which are not required for the processing of the packet at layer L, but may

be important at layer $L+1$.

R_n^L contains all the header information of the $n-1$-st packet (at layers $L-1$, L, and $L+1$) and that of the n-th packet at layer $L-1$

$$R_n^L = \left\{\mathbf{k}_{n-1}^{L-1}, \mathbf{k}_{n-1}^{L}, \mathbf{k}_{n-1}^{L+1}, \mathbf{k}_{n}^{L-1}, \mathbf{p}_{n-1}^{L-1}, \mathbf{p}_{n-1}^{L}, \mathbf{p}_{n-1}^{L+1}, \mathbf{p}_{n}^{L-1}, \mathbf{u}_{n-1}^{L-1}, \mathbf{u}_{n-1}^{L}, \mathbf{u}_{n-1}^{L+1}, \mathbf{u}_{n}^{L-1}\right\}.$$

In addition, data not covered by the CRC at layer L are denoted by $\mathbf{x}_n^L$.

All the bits protected by the CRC are collected in the vector $\mathbf{r}_n^L = \left[\mathbf{k}_n^L, \mathbf{p}_n^L, \mathbf{u}_n^L, \mathbf{o}_n^L\right]$ which contains the above defined fields. Note that the order of the bits in $\mathbf{r}_n^L$ does not correspond to the order in which data are actually transmitted in the n-th packet, but we use this notation for mathematical convenience. The CRC $\mathbf{c}_n^L$ associated to $\mathbf{r}_n^L$ is evaluated as $\mathbf{c}_n^L = \mathcal{F}^L\left(\mathbf{r}_n^L\right)$, where $\mathcal{F}^L$ is a generic encoding function.

When there is no ambiguity, the indices n and L are omitted in what follows.

The evaluation of $\mathbf{c}$ depends on a generator polynomial $g(x) = \sum_{i=0}^{\ell(\mathbf{c})} g_i x^i$ characterizing the CRC [2]. A systematic generator matrix $\mathbf{G} = [\mathbf{I}, \mathbf{\Pi}]$ can be associated to $g(x)$, taking into account the reordering of the bits in $\mathbf{r}$. Using $\mathbf{G}$, $\mathbf{c}$ may be obtained iteratively as follows

$$\begin{cases} \mathbf{c}^0 = \mathbf{0}, \\ \mathbf{c}^{j+1} = \mathcal{F}(\mathbf{r}^{j+1}) = \mathbf{c}^j \oplus \left(r_{j+1} \cdot \boldsymbol{\pi}(j+1)\right). \end{cases} \tag{1}$$

In (1), $\mathbf{r}^j = [r_1 \dots r_j, 0 \dots 0]$, $\oplus$ is the XOR operator, and $\boldsymbol{\pi}(j)$ represents the parity vector associated to bit r_j, which corresponds to the j-th line of $\mathbf{\Pi}$. After $\ell(\mathbf{r})$ iterations, $\mathbf{c}^{\ell(\mathbf{r})} = \mathcal{F}(\mathbf{r}) = \mathbf{c}$.

Assume that the data have been transmitted over an AWGN channel (Gaussian noise of zero mean and variance σ^2), and that soft values are forwarded inside the receiver from each layer to the next one. Noisy data and CRC coming from layer $L-1$ are denoted as $\mathbf{y} = [\mathbf{y}_k, \mathbf{y}_p, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c]$, which includes observations (at PHY layer) or estimations (at other layers) of $\mathbf{k}$, $\mathbf{p}$, $\mathbf{u}$, $\mathbf{o}$, and $\mathbf{c}$.

Since $\mathbf{k}$ and $\mathbf{p}$ are known or may be exactly predicted from the already received data, only $\mathbf{u}$ remains to be estimated. A MAP estimator

$$\widehat{\mathbf{u}}_{\text{MAP}} = \arg\max_{\mathbf{u}} P(\mathbf{u}|\mathbf{k}, \mathbf{p}, R, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c), \tag{2}$$

is thus developed, taking into account the observations $\mathbf{y}$, the knowledge of $\mathbf{k}$, $\mathbf{p}$, and R, as well as the CRC properties. After some derivations, one obtains

$$\widehat{\mathbf{u}}_{\text{MAP}} = \arg\max_{\mathbf{u}} P(\mathbf{u}, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, R). \tag{3}$$

Given that the channel is memoryless and assuming that $\mathbf{o}$ is independent of R, one gets

$$P(\mathbf{u}, \mathbf{o}, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, R) = P(\mathbf{u}|\mathbf{k}, \mathbf{p}, R)P(\mathbf{y}_u|\mathbf{u})P(\mathbf{o}, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, \mathbf{u}). \quad (4)$$

For the sake of generality, assume that $\mathbf{u}$ does not necessarily take all the $2^{\ell(\mathbf{u})}$ values, and that a study of the protocol allows to define $\Omega_u = \Omega_u(\mathbf{k}, \mathbf{p}, R)$, the set of possible values of $\mathbf{u}$. Further assume that these values are equally likely. Thus one may write

$$P(\mathbf{u}|\mathbf{k}, \mathbf{p}, R) = P(\mathbf{u}|\Omega_u) = 1/|\Omega_u|, \quad (5)$$

where $|\Omega_u|$ denotes the cardinal number of Ω_u.

Marginalizing (4) over the $2^{\ell(\mathbf{o})}$ combinations of $\mathbf{o}$, one obtains

$$P(\mathbf{u}, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, R) = P(\mathbf{u}|\Omega_u)P(\mathbf{y}_u|\mathbf{u}) \sum_{\mathbf{o}} P(\mathbf{o}, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, \mathbf{u}), \quad (6)$$

where $\sum_{\mathbf{o}} P(\mathbf{o}, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, \mathbf{u})$ obviously involves the properties of the CRC. Finally, substituting (6) in (3), the MAP estimator is expressed as

$$\widehat{\mathbf{u}}_{\text{MAP}} = \arg \max_{\mathbf{u} \in \Omega_u} P(\mathbf{y}_u|\mathbf{u})\Psi(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_c), \quad (7)$$

with $\Psi(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_c) = \sum_{\mathbf{o}} P(\mathbf{o}, \mathbf{y}_o, \mathbf{y}_c|\mathbf{k}, \mathbf{p}, \mathbf{u})$.

Being very general, the above equations encompass many different situations. For the sake of clarity, the following section details the evaluation of $\widehat{\mathbf{u}}_{\text{MAP}}$ in several practical situations.

IV. Practical Evaluation of the MAP Estimator

A. The set $\mathbf{o}$ is empty

There are many circumstances in which all the bits covered by the CRC belong only to the sets $\mathbf{k}$, $\mathbf{p}$, or $\mathbf{u}$. In these cases, there is no $\mathbf{o}$ and (3) simplifies to

$$\widehat{\mathbf{u}}_{\text{MAP}} = \arg \max_{\mathbf{u} \in \Omega_u} P(\mathbf{y}_u|\mathbf{u}) P\left(\mathbf{y}_c|\mathcal{F}([\mathbf{k}, \mathbf{p}, \mathbf{u}])\right), \quad (8)$$

where $\mathcal{F}([\mathbf{k}, \mathbf{p}, \mathbf{u}])$ is directly evaluated by (1). Hence, an elementary CRC computation replaces the sum over all the possible values of $\mathbf{o}$ and the computational complexity is heavily reduced.

B. The set $\mathbf{o}$ is not empty

When $\mathbf{o}$ is present, we assume that the bits of $\mathbf{o}$ are i.i.d. and do not depend on the other parameters. This is a reasonable approximation since these bits usually depend on the whole corresponding block in the upper layer. The sum in (6) then becomes

$$\Psi(\mathbf{k},\mathbf{p},\mathbf{u},\mathbf{y}_o,\mathbf{y}_c) = \sum_{\mathbf{o}} P(\mathbf{o})P(\mathbf{y}_o|\mathbf{o})P\left(\mathbf{y}_c|\mathcal{F}([\mathbf{k},\mathbf{p},\mathbf{u},\mathbf{o}])\right). \tag{9}$$

Evaluating (9) requires the computation of the sum of probabilities related to the $2^{\ell(\mathbf{o})}$ combinations of $\mathbf{o}$ and to their corresponding CRCs. A direct evaluation has obviously a complexity exponential in $\ell(\mathbf{o})$. This section proposes two methods with reduced complexity: the first one is an exact computation while the second one provides an approximate solution.

C. Exact sum computation - forward method

The CRC can be evaluated iteratively over the data $\mathbf{r}$, as shown by (1). More precisely, the value of the CRC associated to the $j+1$ first bits of $\mathbf{r}$ (shortly, at time $j+1$) only depends on the value of the CRC at time j and on the $j+1$-st bit of $\mathbf{r}$. Each value of the CRC at time j leads to two different values of the CRC at time $j+1$. Consequently, the evolution of the CRC values according to the bits of $\mathbf{r}$ can be described by a trellis. In this trellis, states correspond to the $2^{\ell(\mathbf{c})}$ possible values of the CRC. Transitions are determined by the bits of $\mathbf{r}$. At each time $j = 1 \dots \ell(\mathbf{r})$, we study the contribution of r_j (the j-th bit of $\mathbf{r}$) over the global CRC.

In our case, $\mathbf{r}$ consists of $\mathbf{k}$, $\mathbf{p}$, $\mathbf{u}$, and $\mathbf{o}$. Data contained in $\mathbf{k}$ and $\mathbf{p}$ are assumed to be known, thus have a fixed contribution to the estimate of $\mathbf{u}$. In the forward method, for a given value of $\mathbf{u}$, we want to determine its probability according to the observations $\mathbf{y}$ and the CRC properties, depending on the bits belonging to $\mathbf{o}$. Each value of $\mathbf{u}$, in conjunction with $\mathbf{k}$ and $\mathbf{p}$ defines the initial state in the trellis (there is no contribution from $\mathbf{o}$). Each new bit in $\mathbf{o}$ may provide two new possible states, thus defining a trellis. For any value of $\mathbf{o}$, one gets a path starting from the *same* state associated to $\mathcal{F}([\mathbf{k},\mathbf{p},\mathbf{u},\mathbf{0}])$ and ending in the state associated to $\mathcal{F}([\mathbf{k},\mathbf{p},\mathbf{u},\mathbf{o}])$.

The evaluation of the probability associated to this value of $\mathbf{u}$ is obtained as in (9) by summing the probabilities associated to each combination of $\mathbf{o}$ and the corresponding CRC. These probabilities are classically computed by using the trellis representation. At each time j, a probability is associated to each node and defines the probability of the combinations of $\mathbf{o}^j = [o_1 \dots o_j]$ giving the same value of CRC. Similarly, the vector $\mathbf{y}_o^j$ (observation of $\mathbf{o}^j$)

contains the first j elements of $\mathbf{y}_o$, *i.e.*, $\mathbf{y}_o^j = [y_{o_1} \dots y_{o_j}]$. The remaining information of $\mathbf{o}$, *i.e.*, the $\ell(\mathbf{o}) - j$ last bits, are inserted in $\bar{\mathbf{o}}^j = [o_{j+1} \dots o_{\ell(\mathbf{o})}]$. The vector $\mathbf{y}_{\bar{o}}^j = [y_{o_{j+1}} \dots y_{o_{\ell(\mathbf{o})}}]$ represents the observation of $\bar{\mathbf{o}}^j$. Consequently, $\mathbf{o} = [\mathbf{o}^j, \bar{\mathbf{o}}^j]$ and $\mathbf{y}_o = [\mathbf{y}_o^j, \mathbf{y}_{\bar{o}}^j]$ for all j.

This problem is clearly similar to computing the APPs of the inputs (here parts of inputs) from the measured outputs of block codes. Therefore, our method has many similarities with [21] which deals with soft decoding of block codes. In his paper, Wolf proposes a method based on a trellis, built from the parity check matrix, for the decoding of linear block codes. In our work, the computation is different since the code is divided in three parts: a known portion (vectors $\mathbf{k}$ and $\mathbf{p}$), a candidate value ($\mathbf{u}$), and an unknown part ($\mathbf{o}$ and $\mathbf{c}$). We want to find the best combination of $\mathbf{u}$ by taking into account the redundancy of the code (given by $\mathbf{c}$). The trellis is thus applied to the portions $\mathbf{o}$ and $\mathbf{c}$ for given $\mathbf{k}$, $\mathbf{p}$, and $\mathbf{u}$. Additionally, the technique does not estimate $\mathbf{o}$ and $\mathbf{c}$, but evaluates the probability associated to the $2^{\ell(\mathbf{o})}$ combinations of $[\mathbf{o}, \mathbf{c}]$.

For a given $\mathbf{u}$, this technique allows to evaluate (9) with a complexity $\mathcal{O}\left(\ell(\mathbf{o})2^{\ell(\mathbf{c})}\right)$, compared to $\mathcal{O}\left(2^{\ell(\mathbf{o})}\right)$ with a straightforward computation. Nevertheless, the initial state in the trellis, defined by $\mathcal{F}([\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{0}])$, changes for each possible value of $\mathbf{u} \in \Omega_u$. Consequently, a new trellis has to be constructed for each candidate value $\mathbf{u} \in \Omega_u$. The global complexity of the process is thus $\mathcal{O}\left(|\Omega_u|\ell(\mathbf{o})2^{\ell(\mathbf{c})}\right)$. Example 1 illustrates the trellis constructed for a forward evaluation of $\Psi(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_c)$.

Example 1: The trellis obtained for a systematic binary Hamming$(7, 4)$ code with

$$\mathbf{\Pi} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

is represented in Fig. 3. In this example, we assume that $\ell([\mathbf{k}, \mathbf{p}, \mathbf{u}]) = 1$ bit, $[\mathbf{k}, \mathbf{p}, \mathbf{u}] = [1]$, and $\ell(\mathbf{o}) = 3$ bits. The initial state in the trellis is thus defined by $\mathcal{F}([1, 0, 0, 0]) = [1, 1, 0]$. As explained above, another trellis has to be constructed for evaluating (9) when $[\mathbf{k}, \mathbf{p}, \mathbf{u}] = [0]$. In this case, the initial state changes and is given by $\mathcal{F}([0, 0, 0, 0]) = [0, 0, 0]$. $\diamondsuit$

The next subsection proposes a solution making use of a backward construction of the trellis, which does not require repeated evaluation of metrics on the trellis depending of $\mathbf{u}$.

D. Exact sum computation - backward method

In the following, $V_{\mathbf{s}_i}(j)$ denotes the probability associated to state i at time j in the trellis, for a given $\bar{\mathbf{o}}^j$. $V_{\mathbf{s}_i}(j)$ represents the sum of probabilities associated to each combination of $\bar{\mathbf{o}}^j$ and its corresponding CRC when starting from $\mathbf{s}_i$ at time j

$$V_{\mathbf{s}_i}(j) = \sum_{\bar{\mathbf{o}}^j} P(\bar{\mathbf{o}}^j) P(\mathbf{y}_{\bar{o}}^j | \bar{\mathbf{o}}^j) P\left(\mathbf{y}_c | \mathbf{s}_i \oplus \mathcal{F}([\mathbf{0}, \bar{\mathbf{o}}^j])\right), \tag{10}$$

for $i = 0, 1 \ldots 2^{\ell(\mathbf{c})} - 1$.

Applying (10) to state i at time $j-1$ results in

$$\begin{aligned} V_{\mathbf{s}_i}(j-1) &= \sum_{\bar{\mathbf{o}}^{j-1}} P(\bar{\mathbf{o}}^{j-1}) P(\mathbf{y}_{\bar{o}}^{j-1} | \bar{\mathbf{o}}^{j-1}) P\left(\mathbf{y}_c | \mathbf{s}_i \oplus \mathcal{F}([\mathbf{0}, \bar{\mathbf{o}}^{j-1}])\right) \\ &= P(o_j = 0) P(y_{o_j} | o_j = 0) \sum_{\bar{\mathbf{o}}^j} P(\bar{\mathbf{o}}^j) P(\mathbf{y}_{\bar{o}}^j | \bar{\mathbf{o}}^j) P\left(\mathbf{y}_c | \mathbf{s}_i \oplus \mathcal{F}([\mathbf{0}, \bar{\mathbf{o}}^j])\right) \\ &+ P(o_j = 1) P(y_{o_j} | o_j = 1) \sum_{\bar{\mathbf{o}}^j} P(\bar{\mathbf{o}}^j) P(\mathbf{y}_{\bar{o}}^j | \bar{\mathbf{o}}^j) P\left(\mathbf{y}_c | \mathbf{s}_q \oplus \mathcal{F}([\mathbf{0}, \bar{\mathbf{o}}^j])\right) \\ &= P(o_j = 0) P(y_{o_j} | o_j = 0) V_{\mathbf{s}_i}(j) + P(o_j = 1) P(y_{o_j} | o_j = 1) V_{\mathbf{s}_q}(j), \end{aligned} \tag{11}$$

where $\mathbf{s}_q = \mathbf{s}_i \oplus \boldsymbol{\pi}(j)$. (11) above is the key for computing $V_{\mathbf{s}_i}(j)$ through a backward iteration over the bits of $\mathbf{o}$.

After $\ell(\mathbf{o})$ iterations, $V_{\mathbf{s}_i}(0)$ may be expressed as

$$V_{\mathbf{s}_i}(0) = \sum_{\mathbf{o}} P(\mathbf{o}) P(\mathbf{y}_o | \mathbf{o}) P\left(\mathbf{y}_c | \mathbf{s}_i \oplus \mathcal{F}([\mathbf{0}, \mathbf{o}])\right), \tag{12}$$

for $i = 0, 1 \ldots 2^{\ell(\mathbf{c})} - 1$. This method allows to simultaneously compute (9) for *all* values of $\mathbf{u} \in \Omega_u$. It is no more necessary to construct a new trellis for each value of $\mathbf{u}$. For a given $\mathbf{u}$, the corresponding probability is given by

$$\begin{aligned} V_{\mathcal{F}([\mathbf{k},\mathbf{p},\mathbf{u},\mathbf{0}])}(0) &= \sum_{\mathbf{o}} P(\mathbf{o}) P(\mathbf{y}_o | \mathbf{o}) P\left(\mathbf{y}_c | \mathcal{F}([\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{0}]) \oplus \mathcal{F}([\mathbf{0}, \mathbf{o}])\right) \\ &= \Psi(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_c). \end{aligned}$$

The steps for evaluating (9) backwards are summarized below. The trellis is constructed by starting from $j = \ell(\mathbf{o})$ and going backwards to $j = 0$.

Step 1 - At time $j = \ell(\mathbf{o})$, $V_{\mathbf{s}_i}(\ell(\mathbf{o})) = P(\mathbf{y}_c | \mathbf{s}_i)$ for $i = 0, 1 \ldots 2^{\ell(\mathbf{c})} - 1$.

Step 2 - For $j = \ell(\mathbf{o}) - 1 \ldots 1, 0$, $V_{\mathbf{s}_i}(j)$ is updated according to (11) as

$$V_{\mathbf{s}_i}(j) = P(o_{j+1} = 0) P(y_{o_{j+1}} | o_{j+1} = 0) V_{\mathbf{s}_i}(j+1) + P(o_{j+1} = 1) P(y_{o_{j+1}} | o_{j+1} = 1) V_{\mathbf{s}_q}(j+1),$$

where $i = 0, 1 \ldots 2^{\ell(\mathbf{c})} - 1$ and $\mathbf{s}_q = \mathbf{s}_i \oplus \boldsymbol{\pi}(j+1)$.

Step 3 - After $\ell(\mathbf{o})$ iterations, for any value $\mathbf{u} \in \Omega_u$, $\Psi(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_c) = V_{\mathcal{F}([\mathbf{k},\mathbf{p},\mathbf{u},\mathbf{0}])}(0)$.

With this method, we can directly evaluate (9) for each state i such as $\mathbf{s}_i = \mathcal{F}([\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{0}])$ with $\mathbf{u} \in \Omega_u$. The global complexity of the process is thus $\mathcal{O}\left(\ell(\mathbf{o})2^{\ell(\mathbf{c})}\right)$.

Example 2: Using the same code as in Example 1, one gets the trellis (constructed backwards) of Fig. 4. The sum in (9) is simultaneously computed for the 2 possible initial states $\mathcal{F}([\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{0}]) = [0, 0, 0]$ and $\mathcal{F}([\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{0}]) = [1, 1, 0]$ obtained when $[\mathbf{k}, \mathbf{p}, \mathbf{u}] = [0]$ and $[\mathbf{k}, \mathbf{p}, \mathbf{u}] = [1]$ respectively. $\diamondsuit$

E. Approximate sum computation

Most CRCs are larger than 16 bits and the complexity $\mathcal{O}\left(\ell(\mathbf{o})2^{\ell(\mathbf{c})}\right)$ is too large to allow a real-time implementation of the method presented in Section IV-D. An approximate computation consists in splitting the CRC into m_b blocks of $\ell(\mathbf{c})/m_b$ bits, each block being assumed statistically independent from the others. Thus, $\mathbf{y}_c$ may be written as $\mathbf{y}_c = [\mathbf{y}_{c_1}, \mathbf{y}_{c_2} \dots \mathbf{y}_{c_{m_b}}]$. Using the independence approximation, the sum in (9) becomes

$$\Psi(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_c) \approx \prod_{m=1}^{m_b} \Psi_m(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_{c_m}), \tag{13}$$

with

$$\Psi_m(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_{c_m}) = \sum_{\mathbf{o}} P(\mathbf{o}) P(\mathbf{y}_o|\mathbf{o}) P\left(\mathbf{y}_{c_m}|\mathcal{F}_m([\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{o}])\right), \tag{14}$$

where $\mathcal{F}_m$ is the encoding function associated to the columns $(m-1)\cdot\frac{\ell(\mathbf{c})}{m_b}+1$ to $m\cdot\frac{\ell(\mathbf{c})}{m_b}$ of $\mathbf{\Pi}$, corresponding to a partial CRC of $\ell(\mathbf{c})/m_b$ bits.

The evaluation of (14) is similar to that of Ψ described in Section IV-D. The only difference lies in the size of the trellis: $2^{\ell(\mathbf{c})/m_b}$ states have to be considered at any depth (instead of $2^{\ell(\mathbf{c})}$ states without splitting the CRC). The total complexity for evaluating (13) is now $\mathcal{O}\left(m_b\ell(\mathbf{o})2^{\ell(\mathbf{c})/m_b}\right)$, at the cost of a slightly suboptimal performance.

V. Application to 802.11 Standard

In this paper, we focus on the downlink multimedia transmission over the 802.11 radio interface [15]. First, the format of packets at PHY and MAC layers are briefly recalled in Sections V-A and V-B. Intra-layer and inter-layer redundancy are then described in Section V-C. The resulting processing details for the enhanced permeable layer mechanism are finally proposed in Sections V-D, V-E, and V-F.

A. DSSS PHY layer description

At PHY layer, the 802.11 standard provides 1 or 2 Mbps transmission rate in the 2.4 GHz band using either *Frequency Hopping Spread Spectrum* (FHSS) or *Direct Sequence Spread Spectrum* (DSSS). In DSSS, an 11-chip Barker code sequence is used for spreading the 1 Mbps bitstream. The coded flow thus represents an 11 MHz baseband signal. A DBPSK or DQPSK modulation is applied depending on the required bitrate.

The DSSS PHY packet format is illustrated in Fig. 5. The preamble and the header are transmitted by using the 1 Mbps DBPSK modulation while the payload is modulated either in 1 Mbps DBPSK or 2 Mbps DQPSK. In such PHY packets, the *SYNC* and *SFD* fields consist of 144 known bits, which are not protected by the CRC. These fields are used to estimate the variance of the channel noise (see Section V-F).

The CCITT CRC-16 $\mathbf{c}^{\text{PHY}}$ of 2 bytes protects the *Signal*, *Service*, and *Length* fields; its associated encoding function is denoted by $\mathcal{F}^{\text{PHY}}$. The payload, assumed to contain only one MAC packet, is not protected at this layer. *Service* is reserved for future recommendation. It is set to 00_{16}, and included in $\mathbf{k}^{\text{PHY}}$, according to the notations of Section III. *Signal* indicates the payload modulation and is equal to $0A_{16}$ or 14_{16} for 1 or 2 Mbps bitrate respectively. *Length* indicates on 2 bytes the number of microseconds required to transmit the payload. It depends on both the bitrate and the payload size. It ranges from 16 to $2^{16}-1$. *Signal* and *Length* form thus $\mathbf{u}^{\text{PHY}}$. At this layer, $\mathbf{p}^{\text{PHY}} = \mathbf{o}^{\text{PHY}} = \emptyset$ and $\mathbf{x}^{\text{PHY}}$ contains the $\ell(\mathbf{x}^{\text{PHY}})$ bits of payload.

B. MAC layer description

The MAC packet format is depicted in Fig. 6. In this packet, the CRC $\mathbf{c}^{\text{MAC}}$ of 4 bytes protects both the header fields and the payload; its encoding function is $\mathcal{F}^{\text{MAC}}$.

Considering a non-encrypted downlink transmission of ordered MAC data packets with de-activated retransmission and power-save mode, the 2-byte *Frame Control* field except the *More Frag* flag are assumed to be known. The 6-byte *Receiver Address* field contains the MAC address of the receiver and is thus known. The last field of the MAC header is reserved for local wireless networks and is composed of 6 bytes of zeros in this study. Using the notations of Section III, all the previously mentioned fields may thus be embedded in $\mathbf{k}^{\text{MAC}}$.

The 6-byte *AP Address* field contains the MAC address of the access point AP. This address is transmitted during the medium reservation procedure (RTS-CTS) and may be totally deduced

by the receiver. The 6-byte *Router Address* field corresponds to the MAC address of the router. Assuming that the AP is connected to a single router and that the router address has been already received in other information packets, *Router Address* may also be predicted by the receiver. The 2-byte *Sequence Control* field contains two parameters: a sequence number and a fragment number. The sequence number represents the value of the current IP packet counter. The fragment number indicates the value of the current MAC data packet counter. In this study, packets are transmitted in order and these parameters can be easily determined: the sequence number is incremented by one for each RTS-CTS and the fragment number is incremented by one for each received MAC data packet. *Sequence Control* can be estimated by the receiver. All this predictable fields are represented by $\mathbf{p}^{\text{MAC}}$.

The *More Frag* flag specifies if the current MAC data packet is the last fragment composing an IP packet. The 2 bytes of *Duration* indicate the number of microseconds required to transmit the next MAC fragment and some control packets. Its value depends on the current modulation and the size of the coming MAC data packet. These two fields are embedded in $\mathbf{u}^{\text{MAC}}$. Finally, the payload contains the data to be transmitted and its size is between 0 and 2312 bytes. It is represented by $\mathbf{o}^{\text{MAC}}$.

C. Identifying intra-layer and inter-layer correlations

To evidence these correlations, the transactions at MAC layer have to be described.

In the 802.11 standard, transmission of each IP packet is initialized by a medium reservation procedure at MAC layer consisting of an RTS-CTS exchange. MAC Fragments composing the IP packet are then transmitted to the receiver, which acknowledges them (ACK). In this work, control packets such as RTS, CTS, and ACK are assumed to be correctly received. This assumption is reasonable since these packets are small and DBPSK-modulated. Only errors in data packets (or fragments) will be considered. A *Short Inter-Frame Space* (SIFS) of 10 μs separates two packets successively transmitted over the channel. A *Duration* field is included in each packet and its value indicates the number of microseconds required to transmit the next fragment and some specific packets (CTS and ACK). *Duration* allows to adjust the *Network Allocation Vector* (NAV) for the other terminals. The other stations cannot communicate during the NAV period to avoid interferences.

Assume that D_n^{MAC} and B_n^{PHY} represent the value of *Duration* and the transmission bitrate

(coded in *Signal*) associated to the n-th packet transmitted by the AP (either RTS or data packets). Following the MAC layer specifications of 802.11 standard, D_n^{MAC} is defined as

$$D_n^{\text{MAC}} = 3T_{\text{SIFS}} + 3T_{\text{OVH}} + 2\ell_{\text{C-A}}/B_n^{\text{PHY}} + \ell(\mathbf{x}_{n+1}^{\text{PHY}})/B_n^{\text{PHY}}, \tag{15}$$

except for the last fragment of an IP packet, *i.e.*, when the value of *More Frag* $M_n^{\text{MAC}} = 0$. In this case, one has

$$D_n^{\text{MAC}} = T_{\text{SIFS}} + T_{\text{OVH}} + \ell_{\text{C-A}}/B_n^{\text{PHY}}. \tag{16}$$

In (15) and (16), T_{SIFS} denotes the duration of a SIFS and T_{OVH} represents the duration for transmitting at 1 Mbps the PHY overhead (composed of the preamble and the header of constant size). The other terms depend on the current bitrate B_n^{PHY}. CTS and ACK have the same constant size $\ell_{\text{C-A}}$ and $\ell_{\text{C-A}}/B_n^{\text{PHY}}$ thus corresponds to the duration for sending one of these packets. Finally, $\ell(\mathbf{x}_{n+1}^{\text{PHY}})/B_n^{\text{PHY}}$ refers to the transmission duration of the next PHY payload of $\ell(\mathbf{x}_{n+1}^{\text{PHY}})$ bits.

D. PHY header recovery

For the n-th packet at PHY layer, the observations associated to $\mathbf{k}_n^{\text{PHY}}$, $\mathbf{u}_n^{\text{PHY}}$, and $\mathbf{c}_n^{\text{PHY}}$ defined in Section V-A are collected in $\mathbf{y}_n^{\text{PHY}} = [\mathbf{y}_{k,n}^{\text{PHY}}, \mathbf{y}_{u,n}^{\text{PHY}}, \mathbf{y}_{c,n}^{\text{PHY}}]$. In addition, $\mathbf{y}_{x,n}^{\text{PHY}}$ denotes the observations associated to the $\ell(\mathbf{x}_n^{\text{PHY}})$ bits of the payload $\mathbf{x}_n^{\text{PHY}}$, which is not protected by the CRC.

The number of possible values taken by $\mathbf{u}^{\text{PHY}}$ is significantly reduced when exploiting the *Duration* field contained in the previously received MAC packet (either an RTS or a data packet). Using B_{n-1}^{PHY} and D_{n-1}^{MAC}, one may deduce $\ell(\mathbf{x}_n^{\text{PHY}})$ from (15) as

$$\ell(\mathbf{x}_n^{\text{PHY}}) = \left(D_{n-1}^{\text{MAC}} - 3T_{\text{SIFS}} - 3T_{\text{OVH}} - 2\ell_{\text{C-A}}/B_{n-1}^{\text{PHY}}\right) B_{n-1}^{\text{PHY}}. \tag{17}$$

Then, the duration L_n^{PHY} coded in the *Length* field of the current PHY packet is computed as

$$L_n^{\text{PHY}} = \ell(\mathbf{x}_n^{\text{PHY}})/B_n^{\text{PHY}}. \tag{18}$$

In (17), $\ell(\mathbf{x}_n^{\text{PHY}})$ is totally determined assuming correct estimation of the header of the previous packet. Then, according to (18), L_n^{PHY} may only take two values depending on B_n^{PHY}, which are stored in $\Omega_{u,n}^{\text{PHY}}$. Integrating these properties in (8), one obtains

$$\widehat{\mathbf{u}}_n^{\text{PHY}} = \arg \max_{\mathbf{u}_n^{\text{PHY}} \in \Omega_{u,n}^{\text{PHY}}} P(\mathbf{y}_{u,n}^{\text{PHY}} | \mathbf{u}_n^{\text{PHY}}) P(\mathbf{y}_{c,n}^{\text{PHY}} | \mathbf{c}_n^{\text{PHY}}), \tag{19}$$

with $\mathbf{c}_n^{\text{PHY}} = \mathcal{F}^{\text{PHY}}([\mathbf{k}_n^{\text{PHY}}, \mathbf{u}_n^{\text{PHY}}])$.

E. MAC header recovery

The PHY layer provides $\mathbf{y}_n^{\text{MAC}} = \mathbf{y}_{x,n}^{\text{PHY}} = [\mathbf{y}_{k,n}^{\text{MAC}}, \mathbf{y}_{p,n}^{\text{MAC}}, \mathbf{y}_{u,n}^{\text{MAC}}, \mathbf{y}_{o,n}^{\text{MAC}}, \mathbf{y}_{c,n}^{\text{MAC}}]$ at the input of MAC layer [1]. It contains the observations associated to $\mathbf{k}_n^{\text{MAC}}$, $\mathbf{p}_n^{\text{MAC}}$, $\mathbf{u}_n^{\text{MAC}}$, $\mathbf{o}_n^{\text{MAC}}$, and $\mathbf{c}_n^{\text{MAC}}$ specified in Section V-B.

The number of possible combinations for $\mathbf{u}_n^{\text{MAC}}$ may be significantly reduced when exploiting (15) and (16). Note that D_n^{MAC} is fully determined when $M_n^{\text{MAC}} = 0$. When $M_n^{\text{MAC}} = 1$, the value of *Duration* depends on the next PHY payload size. The number of combinations is associated to the range of MAC payload size. Considering that the payload contains an entire number of bytes, the possible values of $\ell(\mathbf{x}_{n+1}^{\text{PHY}})$ in (15) are given by

$$\ell(\mathbf{x}_{n+1}^{\text{PHY}}) = \ell_{\text{HDR}} + 8i, \tag{20}$$

where $i = 1, 2 \ldots 2312$. In (20), ℓ_{HDR} specifies the known size of the header in a MAC data packet. Then, using (15), (16), and (20), one may show that $\mathbf{u}_n^{\text{MAC}}$ is limited to 2313 combinations which are inserted in $\Omega_{u,n}^{\text{MAC}}$. Combining these properties in (7), one obtains

$$\widehat{\mathbf{u}}_n^{\text{MAC}} = \arg \max_{\mathbf{u}_n^{\text{MAC}} \in \Omega_{u,n}^{\text{MAC}}} P(\mathbf{y}_{u,n}^{\text{MAC}} | \mathbf{u}_n^{\text{MAC}}) \Psi(\mathbf{k}_n^{\text{MAC}}, \mathbf{p}_n^{\text{MAC}}, \mathbf{u}_n^{\text{MAC}}, \mathbf{y}_{o,n}^{\text{MAC}}, \mathbf{y}_{c,n}^{\text{MAC}}), \tag{21}$$

where the second term can be computed with methods presented in Sections IV-D and IV-E.

F. Global scheme

Fig. 7 illustrates the improved permeable layer mechanism applied to the PHY and MAC layers at the receiver, emphasizing on the exchange of information between layers and between consecutive packets, as presented in Sections V-D and V-E.

In addition, we consider that $\mathbf{y}_{s,n}^{\text{PHY}}$ represents the observations of the known preamble $\mathbf{s}^{\text{PHY}}$. As explained in Section V-A, the receiver synchronization is performed with $\mathbf{s}^{\text{PHY}}$. We simultaneously estimate σ^2 from $\mathbf{s}^{\text{PHY}}$ and $\mathbf{y}_{s,n}^{\text{PHY}}$. This measure is essential for working with soft information, as it allows the evaluation of all the likelihoods. The estimator $\widehat{\sigma}^2$ is given by

$$\widehat{\sigma}^2 = \|\mathbf{y}_{s,n}^{\text{PHY}} - \mathbf{s}^{\text{PHY}}\|^2 / \ell(\mathbf{s}^{\text{PHY}}). \tag{22}$$

[1]When encryption is activated, the *WEP* flag in the MAC header is set to 1. In addition, $\mathbf{y}_{o,n}^{\text{MAC}}$ and $\mathbf{y}_{c,n}^{\text{MAC}}$ are the observations of the encrypted bits (plainstream XORed with a pseudo-random keystream). Decryption may easily be performed at receiver side by inverting some LLRs in $\mathbf{y}_{o,n}^{\text{MAC}}$ and $\mathbf{y}_{c,n}^{\text{MAC}}$ according to the known keystream.

Computational complexity is minimized by deactivating the robust header recovery processing when the normal CRC check is successful. It should also be deactivated when the quality of the soft information provided by the lower layer is too poor, *i.e.*, when the signal power is lower than a pre-defined threshold. In such a case, the packet is retransmitted.

VI. Simulation Results

The improved permeable scheme for 802.11 PHY and MAC layers has been implemented (see Fig. 7). A transmission device consisting of a transmitter (AP), an AWGN channel, and a receiver has been simulated using a C program.

The AP generates PHY and MAC packets following the format defined in Section V. The MAC payloads consist of a variable amount of randomly generated bytes. The transmitter modulates data in DBPSK for all the simulations. Three types of header recovery methods are considered at each layer of the receiver. The *standard* decoder performs hard decisions on the data at the channel output. The *robust* decoder exploits only the intra-layer and inter-layer redundancy through a soft decoding algorithm, neglecting the information provided by the CRC. Finally, the *CRC-robust* decoder combines the intra-layer and inter-layer redundancy together with the information provided by the CRC through the soft decoding algorithm presented in Sections V-D and V-E. Performance analysis is performed in terms of HER (Header Error Rate) versus SNR.

In Fig. 8, the standard, robust, and CRC-robust PHY decoders are compared under the assumption that the *Duration* field of the previous MAC packet has been correctly received. Obviously, robust decoders outperform the standard one. An HER of less than 10^{-5} is obtained with the robust decoder for an SNR of 4 dB and with the CRC-robust decoder for an SNR of 2 dB. With the standard decoder, an SNR of at least 15 dB is required to get a comparable HER. At PHY layer, considerable coding gains for a relatively low additional complexity are thus observed, since (8) is used to perform the decoding.

Fig. 9 compares the coding gains obtained by the standard, robust, and CRC-robust MAC decoders. Here, the *Bitrate* field of the current PHY packet is assumed to be correctly received. Two payload sizes (50 and 100 bytes) have been considered. Moreover, the suboptimal method presented in Section IV-E has been used, dividing the CRC in four blocks of 1 byte each. The shape of the curves is very similar to the results obtained at PHY layer, but with significantly smaller gains. Gains due to the MAC CRC information improve with increasing SNR. With

payloads of 100 bytes, HER lower than 10^{-5} are achieved for SNRs of 11 dB, 14 dB, and 15 dB when using CRC-robust, robust, and standard decoders respectively.

Note that the above numbers were obtained under some assumptions (correctly received *Duration* field of the previous MAC packet or *Bitrate* field of the current PHY packet), which allows to study the header recovery mechanism independently at each layer. Our motivation here is to show the large potential interest of such a method.

The MAC processing is more complex than the one done at PHY layer due to the marginalization operation required in (9). The larger the payload, the more complex the decoding process. To reduce the complexity and improve the MAC header recovery performance, the principle of UDP-Lite has been applied at the MAC layer, resulting in a permeable MAC layer (called *MAC-Lite*) where the CRC protects the MAC header field only. In this case, $\mathbf{o}_n^{\text{MAC-L}} = \emptyset$ and (21) becomes

$$\widehat{\mathbf{u}}_n^{\text{MAC-L}} = \arg \max_{\mathbf{u}_n^{\text{MAC-L}} \in \Omega_{u,n}^{\text{MAC-L}}} P(\mathbf{y}_{u,n}^{\text{MAC-L}} | \mathbf{u}_n^{\text{MAC-L}}) P(\mathbf{y}_{c,n}^{\text{MAC-L}} | \mathbf{c}_n^{\text{MAC-L}}), \tag{23}$$

where $\mathbf{c}_n^{\text{MAC-L}} = \mathcal{F}^{\text{MAC-L}}([\mathbf{k}_n^{\text{MAC-L}}, \mathbf{p}_n^{\text{MAC-L}}, \mathbf{u}_n^{\text{MAC-L}}])$.

Standard, robust, and CRC-robust MAC-Lite decoders are depicted in Fig. 10. Comparison with Fig. 9 does not show any difference between MAC and MAC-Lite situations for the standard and robust decoders. This is normal, since the information provided by the CRC is not used by these decoders. However, Fig. 10 demonstrates that the CRC-robust decoder is now significantly more efficient for decoding MAC-lite headers than for decoding classical MAC headers. HER is lower than 10^{-5} for SNRs larger than 3 dB when exploiting the CRC redundancy whereas the two other methods need at least 14 dB. Additionally, the CRC-robust decoding is significantly less complex when processing MAC-Lite headers instead of classical MAC headers, since (23) does not require any marginalization.

Consequently, the combination of the proposed permeable PHY and MAC-Lite layer mechanisms recovers eventually all the PHY and MAC headers from 3 dB SNR onwards. The combination of the proposed permeable PHY and MAC layers reaches this result when the SNR is about 11 dB for an increased complexity. This result demonstrates the potential of replacing the classical MAC layer by the proposed MAC-Lite layer.

VII. Conclusion

A robust header estimation technique has been proposed and has been applied to PHY and MAC layers of WiFi. The main tool of this mechanism consists of a MAP header estimator exploiting jointly the structural properties of the protocol stack along with the CRC redundancy through a soft decoding algorithm. This technique may readily be applied to other layers for various transmission protocols. The estimation technique allows an enhanced permeable layer mechanism (compared, *e.g.*, to UDP-lite) to be defined. This mechanism is particularly well-suited when combined with joint source-channel decoding techniques at Application layer. Simulations with PHY and MAC layers of WiFi illustrate the significant performance gains achieved with the proposed decoding technique. As a result, such techniques allow the headers to be much more robust to channel impairments than the payload, thus avoiding the necessity of packet retransmission in most cases.

References

[1] K. Sayood. *Introduction to Data Compression, Second Edition*. Morgan Kaufmann, San Francisco, 2000.

[2] R. E. Blahut. *Theory and Practice of Error Control Codes*. Addison-Wesley, Reading, MA, 1984.

[3] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison Wesley, Boston, third edition, 2005.

[4] T. Richardson and U. Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008.

[5] M. C. Hong, H. Schwab, L. P. Kondi, and A. K. Katsaggelos. Error concealment algorithms for compressed video. *Signal Processing: Image Communication*, 14:473–492, 1999.

[6] W.-Y. Kung, C.-S. Kim, and C.-C. J. Kuo. Spatial and temporal error concealment techniques for video transmission over noisy channels. *IEEE trans. on circuits and systems for video technology*, 16:789–802, 2006.

[7] V. Buttigieg and P.G. Farrell. A MAP decoding algorithm for variable-length error-correcting codes. In *Codes and Cyphers: Cryptography and Coding IV*, pages 103–119, Essex, England, 1995. The Inst. of Mathematics and its Appl.

[8] S. Kaiser and M. Bystrom. Soft decoding of variable-length codes. In *Proc. IEEE ICC*, volume 3, pages 1203–1207, New Orleans, 2000.

[9] R. Thobanen and J. Kliewer. Robust decoding of variable-length encoded markov sources using a three-dimensional trellis. *IEEE Communications Letters*, 7(7):320–322, 2003.

[10] H. Nguyen, P. Duhamel, J. Brouet, and D. Rouffet. Robust VLC sequence decoding exploiting additional video stream properties with reduced complexity. In *Proc. IEEE ICME*, pages 375–378, June 2004. Taipei, Taiwan.

[11] C. Bergeron and C. Lamy-Bergot. Soft-input decoding of variable-length codes applied to the H.264 standard. In *Proc. IEEE 6th Workshop on Multimedia Signal Processing*, pages 87–90, 29 Sept.-1 Oct. 2004.

[12] C.M. Lee, M. Kieffer, and P. Duhamel. Soft decoding of VLC encoded data for robust transmission of packetized video. In *Proc. IEEE ICASSP*, pages 737–740, 2005.

[13] R. Bauer and J. Hagenauer. On variable length codes for iterative source/channel decoding. In *Proc. IEEE Data Compression Conference*, pages 272–282, Snowbird, UT, 1998.

[14] H. Jenkac, T. Stockhammer, and W. Xu. Permeable-layer receiver for reliable multicast transmission in wireless systems. In *Proc. IEEE Wireless Communications and Networking Conference*, volume 3, pages 1805–1811, 13-17 March 2005.

[15] ANSI/IEEE. 802.11, part 11 : Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Technical report, 1999.

[16] L. A. Larzon, M. Degermark, L. E. Jonsson, and G. Fairhurst. The lightweight user datagram protocol (UDP-Lite). Technical Report RFC 3828, The Internet Society, 2004.

[17] M. Van der Schaar and S. Shankar. Cross-layer wireless multimedia transmission: Challenges, principles, and new paradigms. *IEEE Wireless Communications Magazine*, 12(4):50–58, 2005.

[18] C. Bormann (Ed.). Robust header compression (ROHC): Framework and four profiles. Technical Report RFC 3095, 2001.

[19] J. W. Nieto and W. N. Furman. Cyclic redundancy check (CRC) based error method and device. US Patent US 2007/0192667 A1, Aug. 16 2007.

[20] C. Marin, P. Duhamel, K. Bouchireb, and M. Kieffer. Robust video decoding through simultaneous usage of residual source information and MAC layer CRC redundancy. In *Proc. IEEE Globecom 07*, pages 2070–2074, 2007.

[21] J. K. Wolf. Efficient maximum-likelihood decoding of linear block codes using a trellis. *IEEE Trans. Inform. Theory*, 24(1):76–80, 1978.

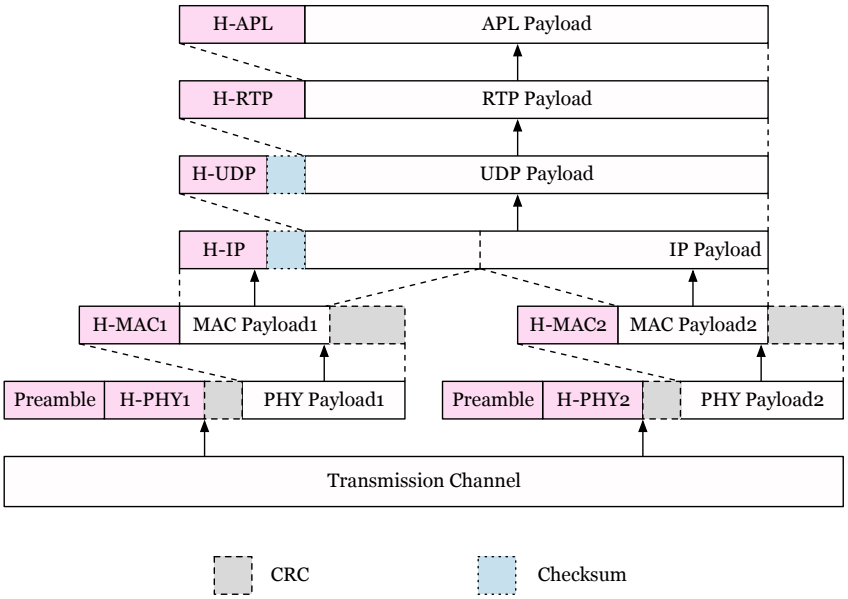


Fig. 1. Protocol stack for multimedia transmission over WiFi

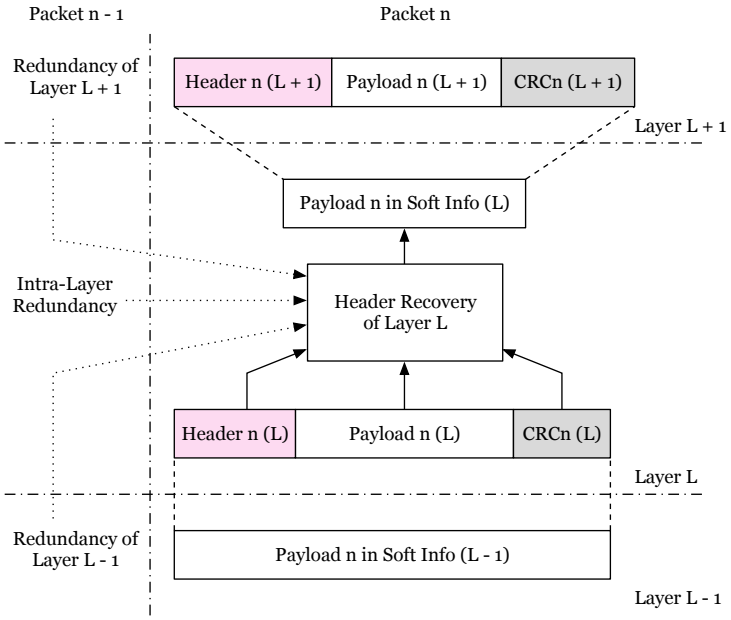


Fig. 2. Proposed permeable layer mechanism

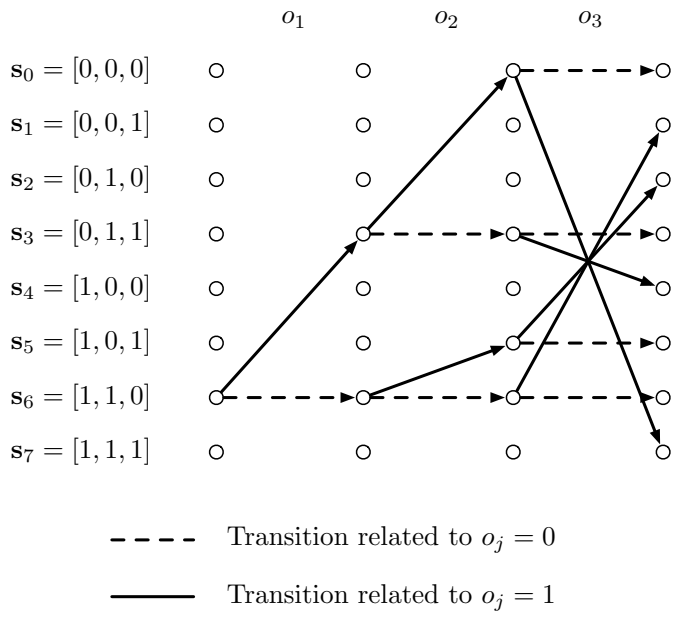


Fig. 3. Trellis obtained with the forward construction

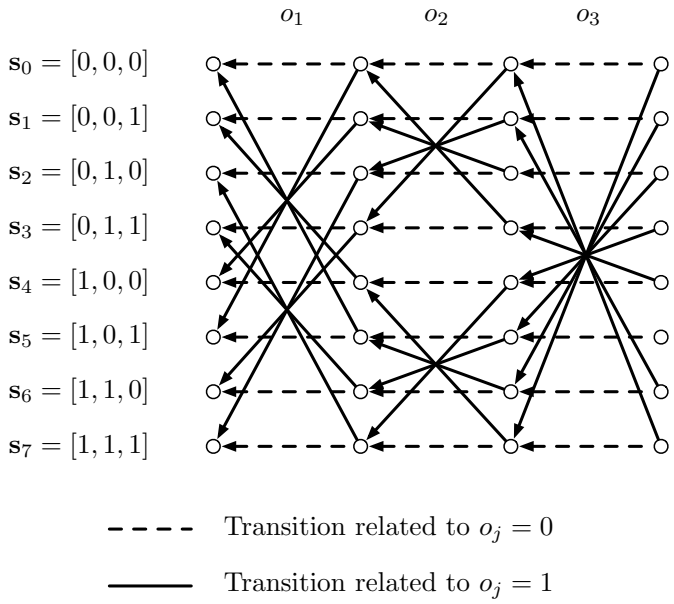


Fig. 4. Trellis obtained with the backward construction

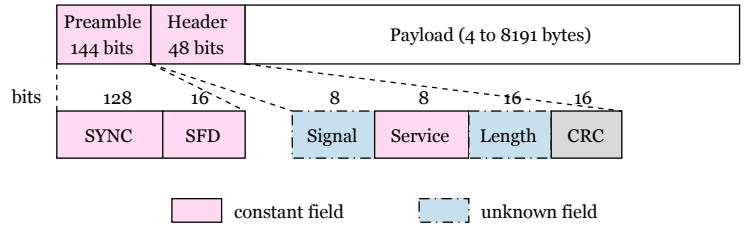


Fig. 5. PHY packet format in 802.11 standard

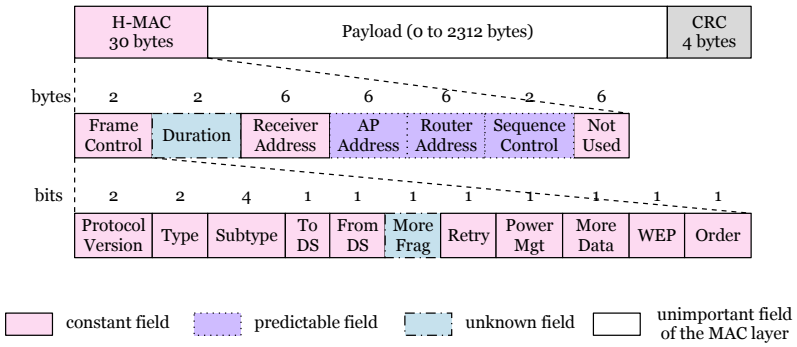


Fig. 6. MAC packet format in 802.11 standard

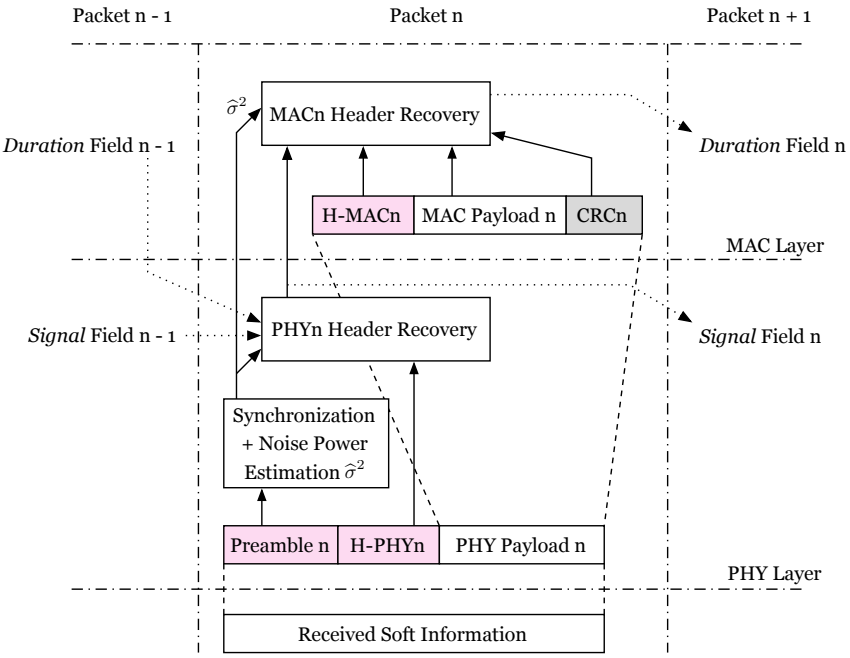


Fig. 7. Proposed scheme for PHY and MAC layers

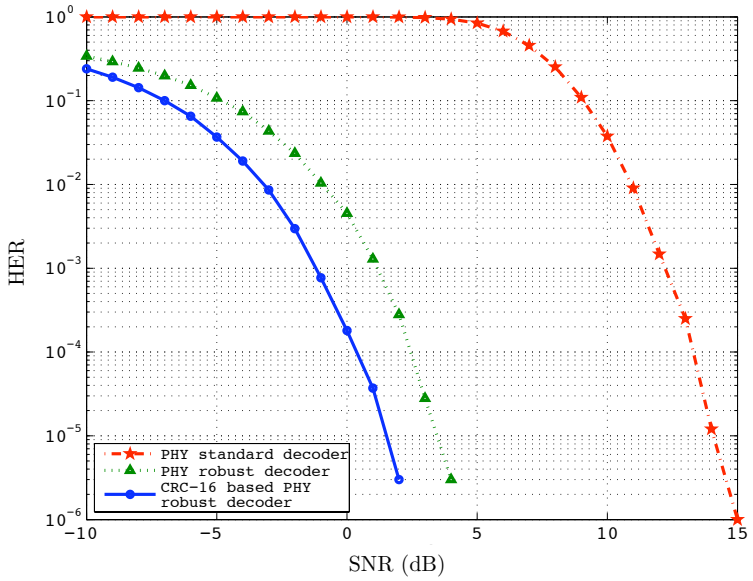


Fig. 8. HER vs SNR for the three PHY decoders

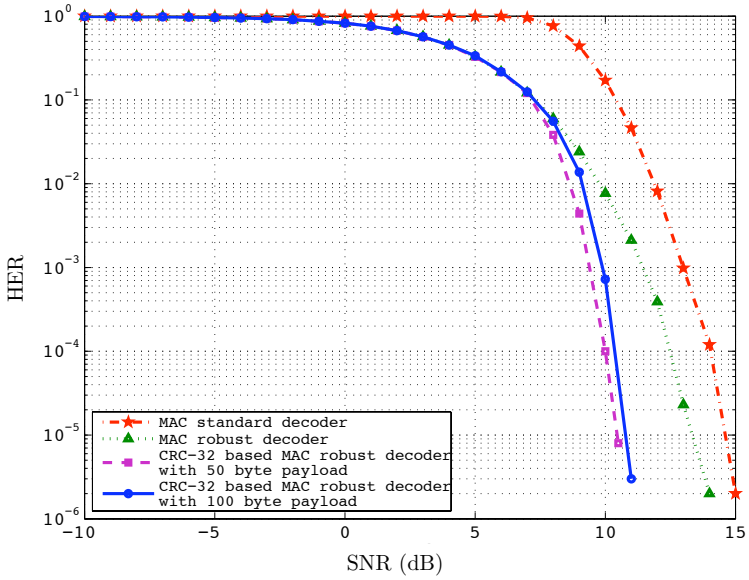


Fig. 9. HER vs SNR for the three MAC decoders

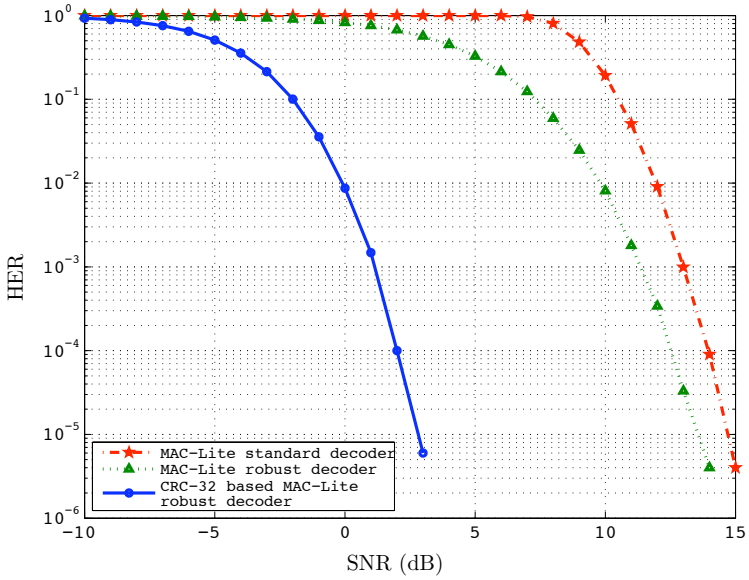


Fig. 10. HER vs SNR for the three MAC-Lite decoders